# MS EXCEL ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΛΟΓΙΣΤΙΚΩΝ ΦΥΛΛΩΝ

Αλέξανδρος Μπάτσης

# Ενότητες

1. Έναρξη – Περιεχόμενο και Στόχοι Προγράμματος
2. Εισαγωγή στις Μακροεντολές
3. Το Περιβάλλον της VBA του Excel
4. Υπορουτίνες
5. Τύποι Δεδομένων & Μεταβλητές
6. Εργασία με κελιά και περιοχές κελιών
7. Εισαγωγή στο Excel Object Model
8. Συναρτήσεις – Συναρτήσεις Οριζόμενες από το Χρήστη
9. Αλληλεπίδραση Εφαρμογής με το Χρήστη
10. Λήψη αποφάσεων στη VBA
11. Οι βρόγχοι στη VBA
12. Συμβάντα
13. Εργασία με Βιβλία και Φύλλα Εργασίας
14. Αποσφαλμάτωση
15. Σύνδεση με Βάσεις Δεδομένων
16. Συνεργασία με άλλες εφαρμογές του Office
17. Στοχευμένη Μελέτη Περίπτωσης στο Δημόσιο Τομέα
18. Αξιολόγηση Προγράμματος
19. Αξιολόγηση Εκπαιδευομένων

# 1. Έναρξη – Περιεχόμενο και Στόχοι του Προγράμματος

Εκπαιδευτικοί Στόχοι:

1. Προσδιορισμός το περιεχόμενο και τους στόχους του επιμορφωτικού προγράμματος
2. Αντίληψη των αναμενόμενων αποτελεσμάτων και της σκοπιμότητας του επιμορφωτικού προγράμματος

# Έναρξη – Περιεχόμενο και Στόχοι του Προγράμματος

- Σύντομη παρουσίαση του Φορέα υλοποίησης του προγράμματος ΙΝ.ΕΠ-ΕΚΔΔΑ
- Σκοπιμότητα σχεδιασμού και υλοποίησης του επιμορφωτικού προγράμματος
- Στόχοι, περιεχόμενο και διάρθρωση του επιμορφωτικού προγράμματος

# 2. Εισαγωγή στις Μακροεντολές

Εκπαιδευτικοί Στόχοι:

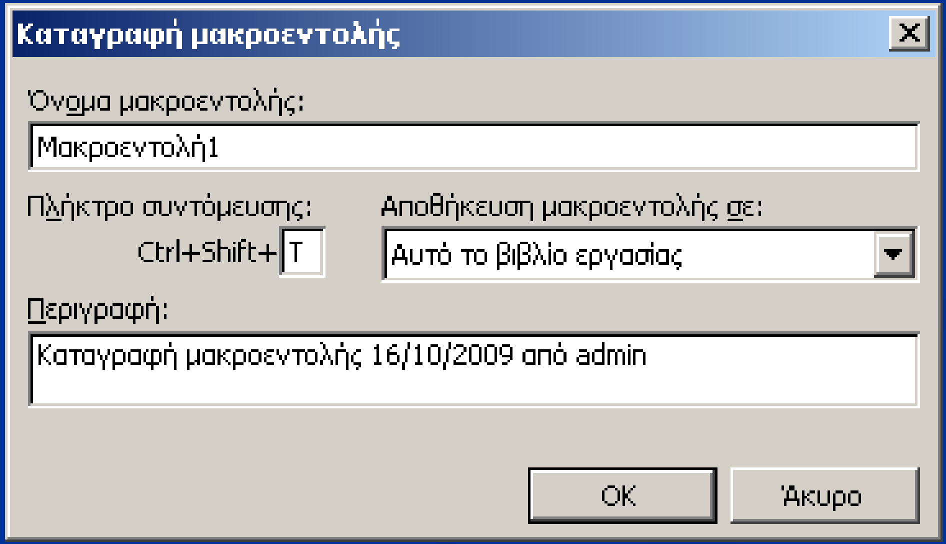
1. Προσδιορισμός της φύσης και χρησιμότητας των μακροεντολών
2. Δημιουργία και εκτέλεση απλών μακροεντολών
3. Σύνδεση μακροεντολών με πλήκτρα

# Τι είναι οι Μακροεντολές

- Σειρές εντολών ή ενεργειών που επαναλαμβάνουμε συχνά
- Ανάγκη αυτοματοποίησης και ομαδοποίησης των ενεργειών αυτών
- Δημιουργία δικής μας εντολής (που αποτελείται από συνδυασμό εντολών του Excel)
- Εντολές κώδικα VBA (Visual Basic for Applications)
- Καταγραφή μακροεντολών από το μενού

# Καταγραφή Μακροεντολών

- [Εργαλεία]→[Μακροεντολή]→[Καταγραφή νέας μακροεντολής]
  - Όνομα Μακροεντολής
  - Πλήκτρο συντόμευσης
  - Αποθήκευση μακροεντολής
  - Περιγραφή/Σχόλια για τη μακροεντολή



# Παράδειγμα

- [Εργαλεία]→[Μακροεντολή]→[Καταγραφή νέας μακροεντολής]
- Όνομα: Mesos_Oros
- Πλήκτρο συντόμευσης: Ctrl + α
- Πατήστε Alt+F11 και παρατηρήστε τον κώδικα
- Αφαιρέστε τη γραμμή με τον κώδικα Range("B1").Select γιατί η μακροεντολή θα πηγαίνει πάντα στο κελί B1
- Δοκιμάστε με τους διπλανούς αριθμούς:
- Επαναλάβετε οπουδήποτε με άλλους αριθμούς

| | A | B | C | D |
|---|---|---|---|---|
| 1 | 2 | =SUM(A1:A5)/COUNT(A1:A5) | | |
| 2 | 1 | | | |
| 3 | 2,5 | | | |
| 4 | 8 | | | |
| 5 | -3 | | | |

| | A |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | -3 |
| 5 | 0 |

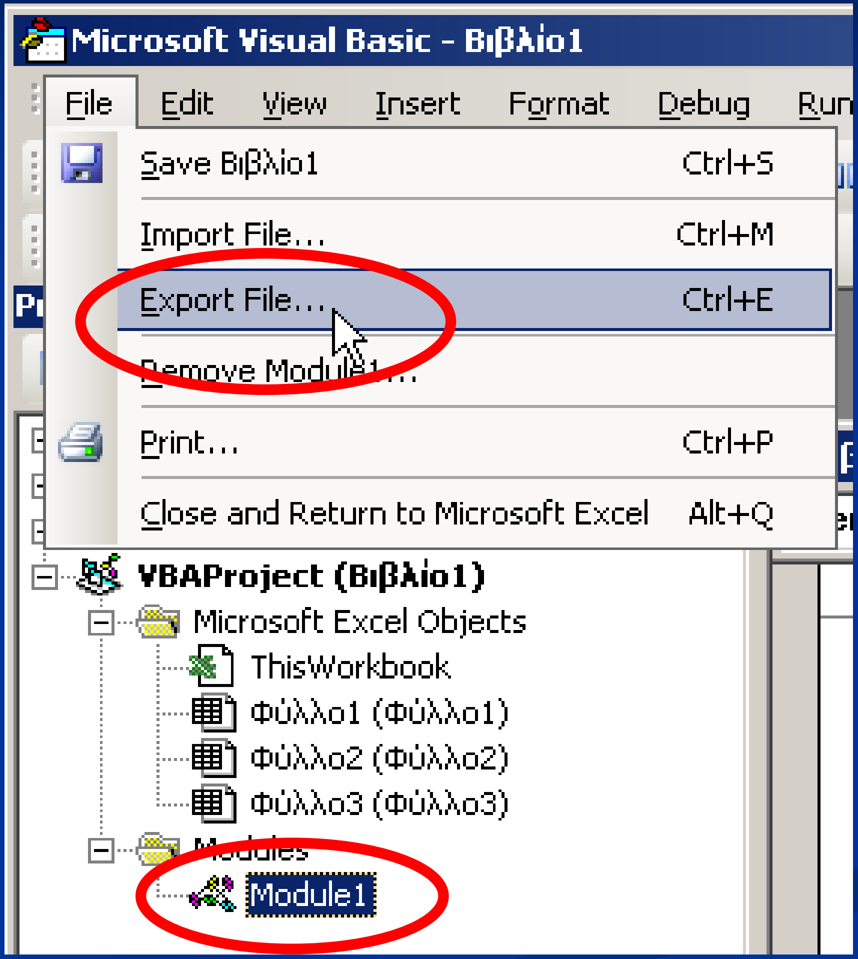
# Άσκηση

- Φτιάξτε τον διπλανό πίνακα
- Δημιουργήστε μία νέα μακροεντολή με όνομα Nees_Stiles και πλήκτρο συντόμευσης Ctrl+β η οποία θα εισάγει δύο νέες στήλες μεταξύ των στηλών «Δ/ΝΣΗ» και «ΤΗΛ.» και θα ονομάζει τις στήλες αυτές «ΠΟΛΗ» και «ΤΚ».

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **ΟΝΟΜΑ** | **ΕΠΩΝΥΜΟ** | **Δ/ΝΣΗ** | **ΤΗΛ.** |
| 2 | ΟΝ1 | ΕΠ1 | ΔΣ1 | ΤΗΛ1 |
| 3 | ΟΝ2 | ΕΠ2 | ΔΣ2 | ΤΗΛ2 |
| 4 | ΟΝ3 | ΕΠ3 | ΔΣ3 | ΤΗΛ3 |
| 5 | ΟΝ4 | ΕΠ4 | ΔΣ4 | ΤΗΛ4 |
| 6 | ΟΝ5 | ΕΠ5 | ΔΣ5 | ΤΗΛ5 |
| 7 | ΟΝ6 | ΕΠ6 | ΔΣ6 | ΤΗΛ6 |
| 8 | ΟΝ7 | ΕΠ7 | ΔΣ7 | ΤΗΛ7 |
| 9 | ΟΝ8 | ΕΠ8 | ΔΣ8 | ΤΗΛ8 |
| 10 | ΟΝ9 | ΕΠ9 | ΔΣ9 | ΤΗΛ9 |
| 11 | ΟΝ10 | ΕΠ10 | ΔΣ10 | ΤΗΛ10 |

# Αποθήκευση Μακροεντολής

- Αποθήκευση μακροεντολής ως αρχείο Visual Basic
- [File]→[Export File…]
- Και αποθήκευση ως MesosOros.bas
- Προσέξτε να είναι επιλεγμένο το Module1

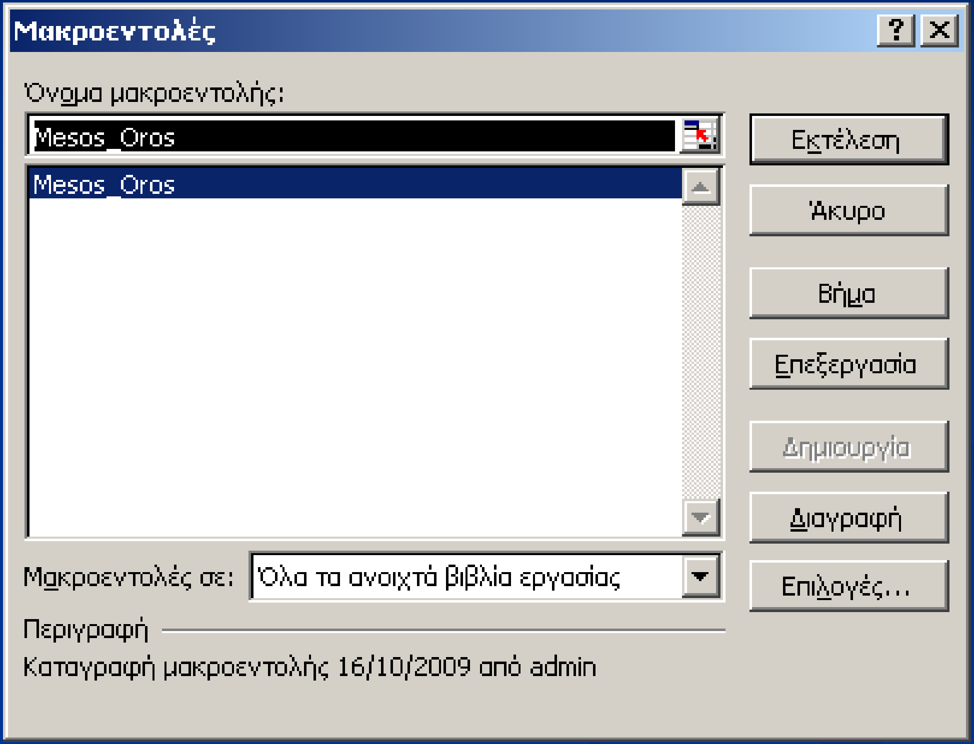


# Παράδειγμα
## (Φόρτωση αποθηκευμένης μακροεντολής)

- Ξεκινήστε με ένα νέο βιβλίο εργασίας
- Πατήστε [Alt+F11] και επιλέξτε [File]→[Import File…]
- Εισάγετε το αρχείο MesosOros.bas
- Δοκιμάστε για την ορθή εκτέλεση της μακροεντολής

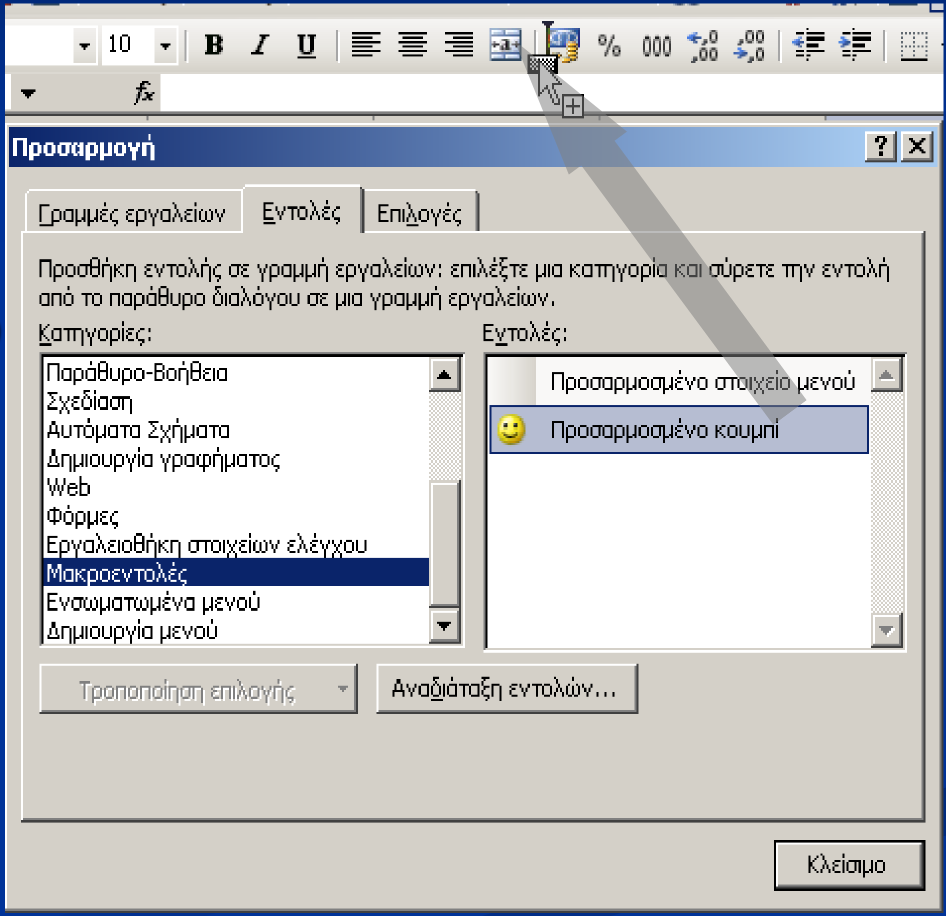
# Επιλογές Μακροεντολών

- [Εργαλεία]→[Μακροεντολή] →[Μακροεντολές]
- ή [Alt+F8]
- Εκτέλεση
- Βήμα (βηματική εκτέλεση)
- Επεξεργασία
- Δημιουργία νέας
- Διαγραφή
- Επιλογές (αλλαγή πλήκτρου συντόμευσης)



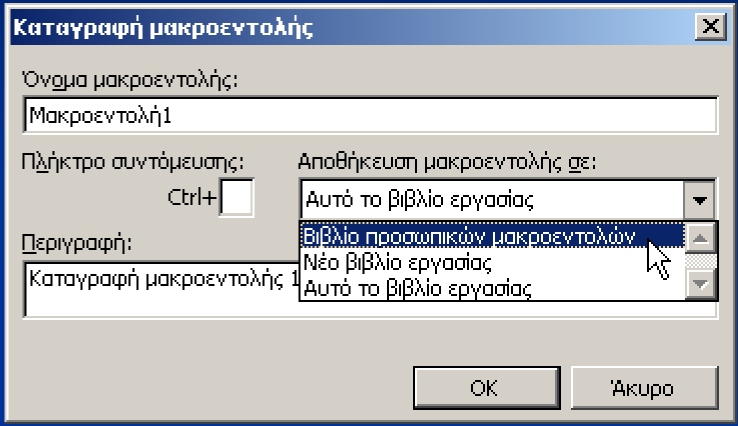
# Εκτέλεση μακροεντολής από εικονίδιο στη γραμμή εργαλείων

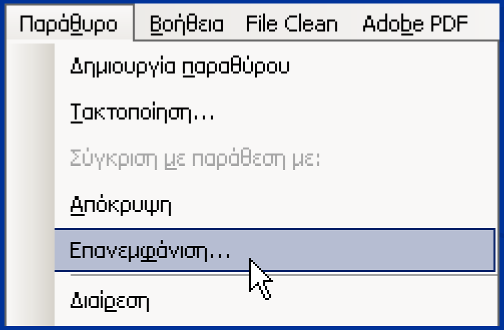
- [Εργαλεία]→[Προσαρμογή]
- Καρτέλα [Εντολές]
- Κατηγορία: Μακροεντολές
- Σύρουμε το «Προσαρμοσμένο κουμπί» στη γραμμή εργαλείων που θέλουμε
- Δεξί κλικ στο εικονίδιο και «Αντιστοίχηση μακροεντολής»

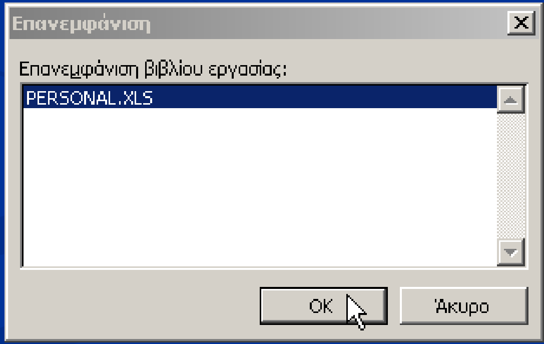


# Βιβλίο προσωπικών μακροεντολών (personal.xls)

- Για να εκτελούνται οι μακροεντολές μας από οποιοδήποτε βιβλίο εργασίας
- Το personal.xls είναι κρυφό
- Για να επεξεργασία, διαγραφή μακροεντολών πρέπει να εμφανίσουμε το personal.xls



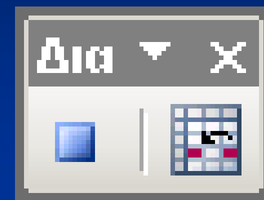



# Αυτόματη εκτέλεση μακροεντολής κατά το άνοιγμα του βιβλίου εργασίας

- Αποθηκεύουμε τη μακροεντολή με το όνομα Auto_Open
- Εμποδίζουμε την εκτέλεση της μακροεντολής Auto_Open αν κατά το άνοιγμα του βιβλίου εργασίας κρατάμε πατημένο το πλήκτρο Shift

# Απόλυτες και σχετικές αναφορές κατά την καταγραφή μακροεντολής

- Όταν θέλουμε με μία μακροεντολή να εισάγουμε στα κελιά A1 ως A12 τους 12 μήνες χρησιμοποιούμε απόλυτες αναφορές
- Αν θέλουμε να εισάγουμε τους μήνες στο ενεργό κελί και στα κελιά κάτω από αυτό τότε χρησιμοποιούμε σχετικές αναφορές




Καταγραφή Μακροεντολής με Απόλυτες αναφορές




Καταγραφή Μακροεντολής με Σχετικές αναφορές

# Άσκηση: Σχετικές/Απόλυτες Αναφορές

- Δημιουργήστε τον διπλανό κώδικα
- Ο υπολογισμός της δόσης του δανείου προκύπτει από τη συνάρτηση: =PMT(B7/12;B8*12;-B5)
- Κάντε κλικ πάνω στο κελί A3 για να το επιλέξετε
- Δημιουργείστε μία μακροεντολή με όνομα ApolitiAnafora που να μορφοποιεί το κελί B3 σε έντονους (bold) και πλάγιους (italics) χαρακτήρες
- Δημιουργήστε μία άλλη μακροεντολή με όνομα SxetikiAnafora που να κάνει ακριβώς το ίδιο με την προηγούμενη μακροεντολή αλλά με σχετική αναφορά στο διπλανό κελί (δηλαδή να εκτελείται για κάθε διπλανό κελί από το επιλεγμένο)



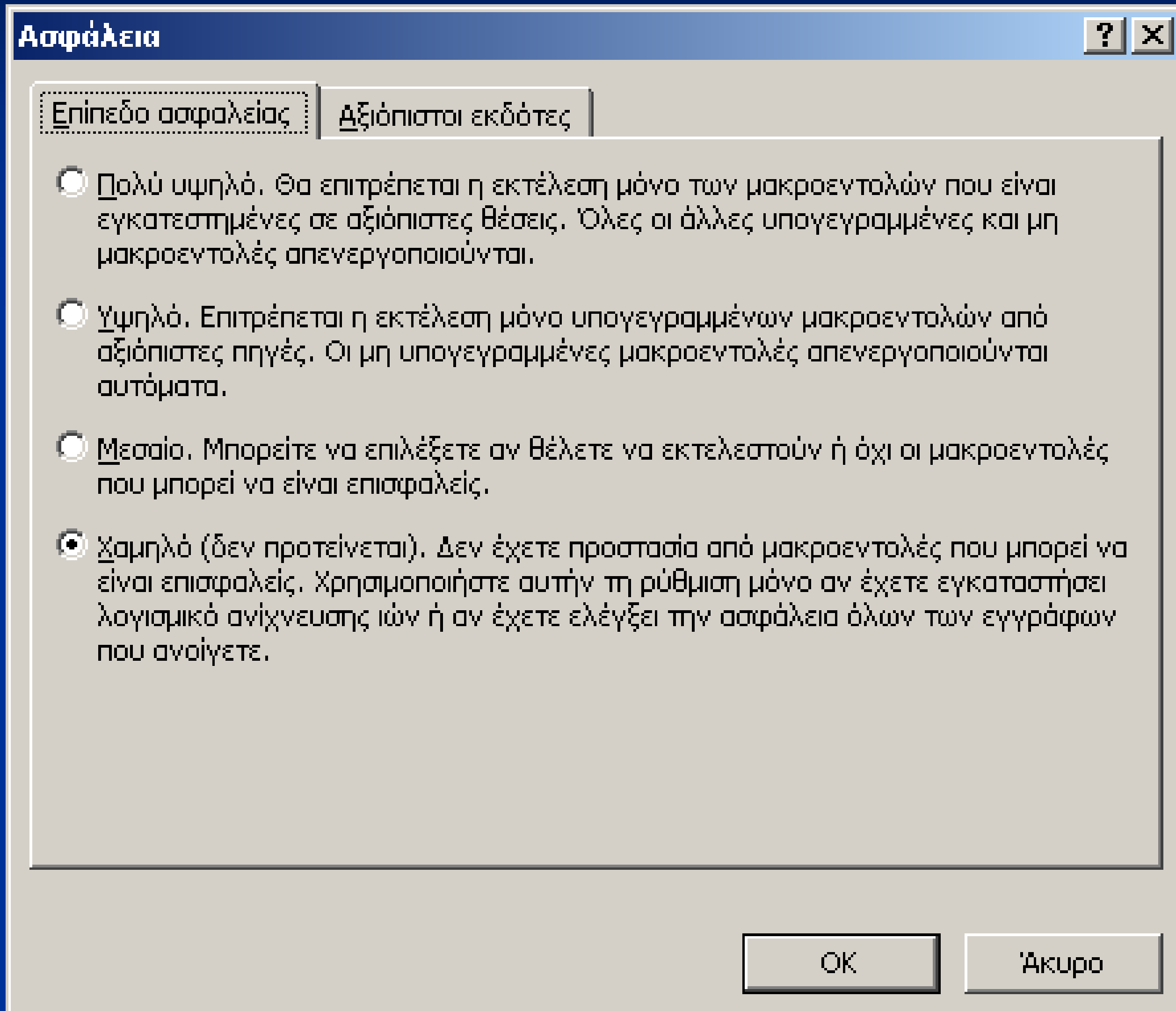




| | A | B |
|---|---|---|
| 1 | **Αγορά Ακινήτου** | |
| 2 | | |
| 3 | Τιμή Ακινήτου | 120.000,00 € |
| 4 | Προκαταβολή | 20.000,00 € |
| 5 | | 100.000,00 € |
| 6 | | |
| 7 | Επιτόκιο | 4,50% |
| 8 | Έτη | 25 |
| 9 | | |
| 10 | Δόση Δανείου (μηνιαία) | 555,83 € |

# Ρύθμιση επιπέδου ασφαλείας μακροεντολών

- [Εργαλεία]→Μακροεντολή]→[Ασφάλεια]

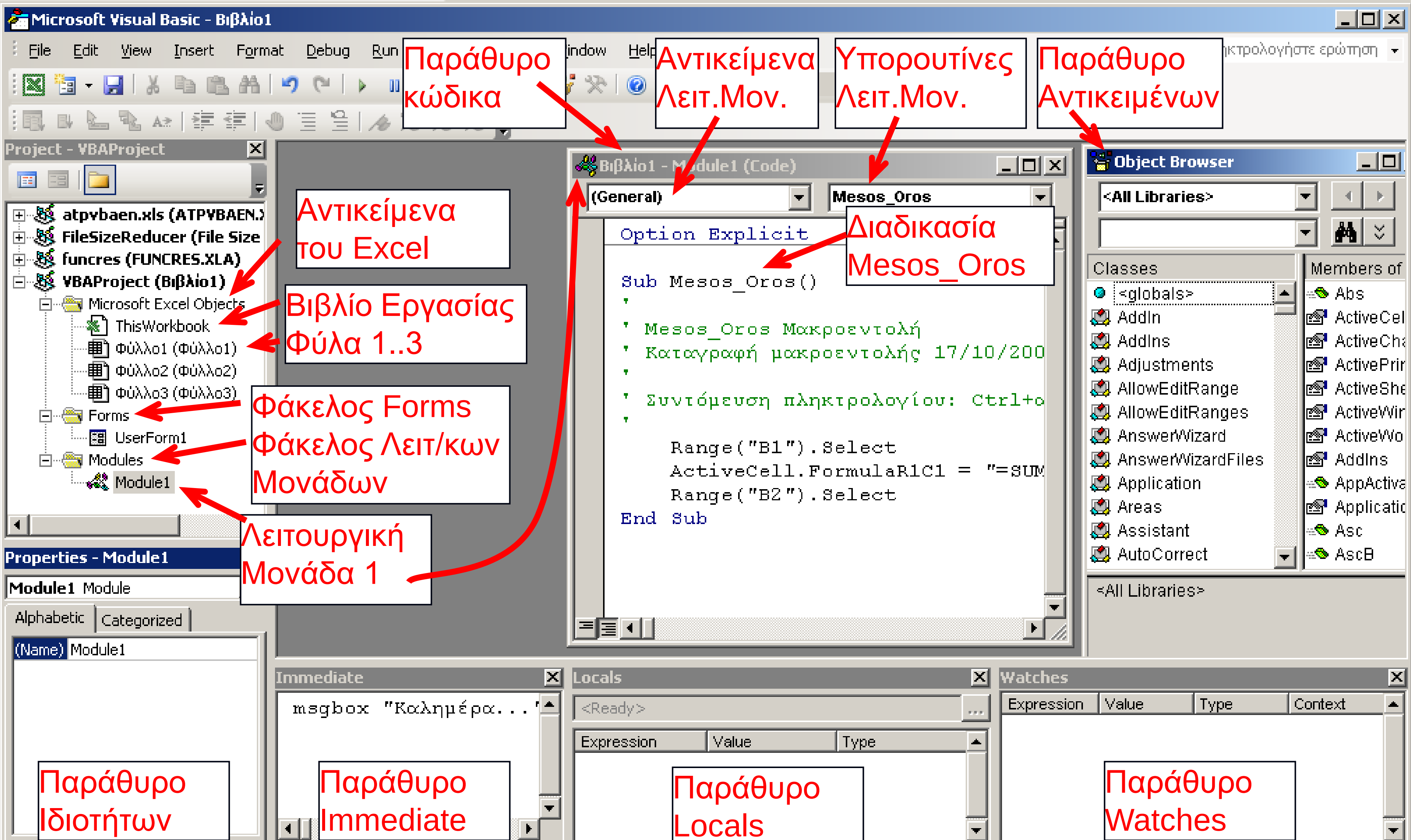


# 3. Το περιβάλλον της VBA του Excel

Εκπαιδευτικοί Στόχοι:

- Περιγραφή της δομής ενός έργου της VBA
- Αναγνώριση της σχέσης μίας μακροεντολής με κώδικα γραμμένο σε VBA
- Ρύθμιση των ιδιοτήτων ενός έργου VBA
- Χειρισμός των modules ενός έργου

# Το περιβάλλον της VBA του Excel



# Τα Συστατικά της VBA

- **Κώδικας** (code). Ο κώδικας αποθηκεύεται σε λειτουργικές μονάδες (modules), σε Φύλλα εργασίας ή σε Φόρμες
- **Λειτουργικές μονάδες** (modules). Περιέχουν τμήματα κώδικα μέσα σε διαδικασίαες (procedures)
- **Διαδικασίες** (procedures). Είναι μονάδες κώδικα που εκτελούν κάποια λειτουργία. Υπάρχουν δύο είδη διαδικασιών:
  - **Υπορουτίνες** (Subroutines): Οι μακροεντολές αποθηκεύονται σαν υπορουτίνες. Μπορούν να κληθούν από άλλες διαδικασίες
  - **Συναρτήσεις** (Functions): Εκτελούν υπολογισμούς και επιστρέφουν πάντα μία τιμή. Έχουν ορίσματα που χρησιμοποιούν για την εκτέλεση των υπολογισμών.

```
Sub Demo()
    Answer = 2 * 5
    Cells(1, 1) = Answer
End Sub
```

```
Function Mult(arg1, arg2)
    Mult = arg1 * arg2
End Function
```

# Τα Συστατικά της VBA

- **<u>Αντικείμενα</u>** (objects): Η VBA χειρίζεται αντικείμενα του Excel όπως: Application, Workbook, Worksheet, Chart, Range κ.α. Τα αντικείμενα μπορούν να περιέχουν άλλα αντικείμενα, π.χ. το αντικείμενο Application περιέχει το αντικείμενο Workbook
- **<u>Συλλογές</u>** (collections): Συλλογές από όμοια αντικείμενα, π.χ. τα φύλλα ενός βιβλίου εργασίας που το καθένα είναι ένα αντικείμενο Worksheet απαρτίζουν τη συλλογή Worksheets. Οι συλλογές είναι και αυτές αντικείμενα.
- **<u>Ιεραρχία αντικειμένων</u>** (object hierarchy): Τα αντικείμενα δομούνται σε μία συγκεκριμένη ιεραρχία, π.χ. αναφερόμαστε στο αντικείμενο «Φύλλο1» ως εξής: Application.Workbooks("Βιβλίο1.xls").Worksheets("Φυλλο1")

# Τα Συστατικά της VBA

- **<u>Ιδιότητες αντικειμένων</u>** (object properties): Τα αντικείμενα έχουν ιδιότητες, οι οποίες έχουν τιμές, π.χ. το αντικείμενο Range, που αντιστοιχεί σε ένα κελί ή περιοχή κελιών, έχει τις ιδιότητες Value και Name. Αναφερόμαστε σε μία ιδιότητα ως εξής: Worksheets(“Φυλλο1”).Range(“A1”).Value = 100

- **<u>Μέθοδοι αντικειμένων</u>** (object methods): Ενέργειες που εκτελούνται με αυτά, π.χ. το αντικείμενο Range έχει τη μέθοδο ClearContents η οποία διαγράφει τα περιεχόμενα του κελιού ή της περιοχής κελιών. Η μέθοδος εκτελείται ως εξής: Range(“A1”).ClearContents
  - Μία μέθοδος μπορεί να έχει μία ή περισσότερες παραμέτρους.
  - Αν έχει μία μόνο παράμετρο τότε η σύνταξη είναι:
    - ObjectName.MethodName parametename = value
    - Π.χ η μέθοδος Delete του αντικειμένου Range έχει ως παράμετρο την Shift που της δίνουμε την τιμή xlShiftUp:
    - Range(“B3:B4”).Delete Shift:=xlShiftUp
  - Αν έχει περισσότερες παραμέτρους μπορούμε να δώσουμε τιμή σε κάποιες ή σε όλες τις παραμέτρους. Αν θέλουμε να δώσουμε τιμή σε μερικές μόνο χρησιμοποιούμε τη σύνταξη:
    - Workbooks.Open(Filename:=“text.xls”, ReadOnly:=True)

# Τα Συστατικά της VBA

- **<u>Συμβάντα</u>** (events): Εκτέλεση κώδικα όταν συμβαίνουν συγκεκριμένα γεγονότα στο Excel, π.χ., όταν ανοίγουμε ένα βιβλίο εργασίας, όταν αλλάζει ένα φύλλο εργασίας, πριν κλείσουμε το βιβλίο εργασίας. Ο κώδικας βρίσκεται σε διαδικασίες με συγκεκριμένα ονόματα π.χ. BeforeClose. Τα συμβάντα συνδέονται με αντικείμενα και κάθε είδος αντικειμένου υποστηρίζει διαφορετικά συμβάντα.
- **<u>Μεταβλητές</u>** (variables): Οι μεταβλητές είναι οντότητες, θέσεις μνήμης του Η/Υ, που χαρακτηρίζονται από ένα μοναδικό όνομα και τον τύπο δεδομένων που αποθηκεύεται. Στο προηγούμενο παράδειγμα της διαδικασίας Demo() η Answer ήταν μεταβλητή.

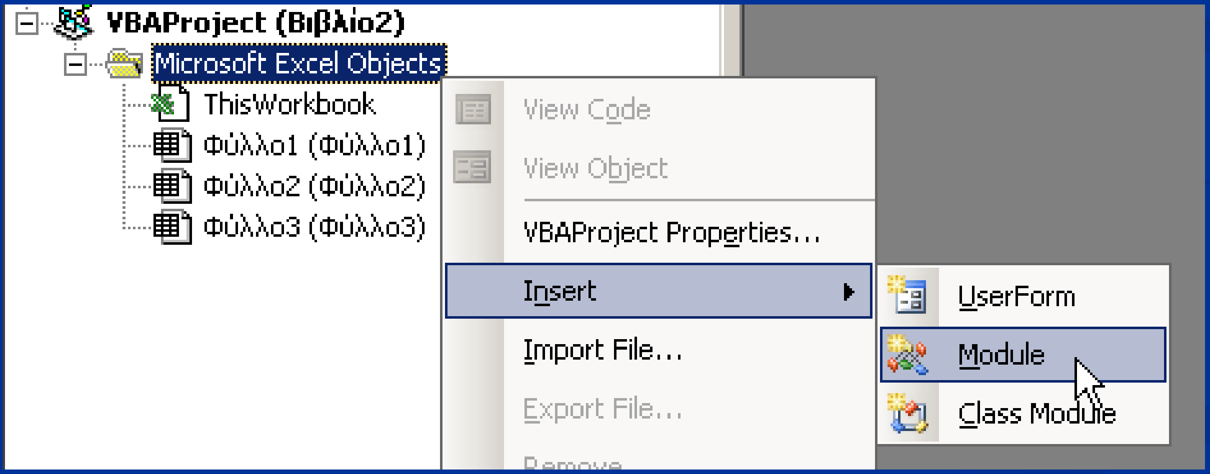
# Λειτουργικές Μονάδες (modules)

- Οι λειτουργικές μονάδες περιέχουν κώδικα στις παρακάτω μορφές:
  - Δηλώσεις (declarations): Χρησιμοποιούνται για να ορίσουμε τον τύπο δεδομένων που αποθηκεύει μία μεταβλητή, ή να ορίσουμε κάποιες παραμέτρους της λειτουργικής μονάδας
  - Υπορουτίνες (Sub): Κώδικας VBA που εκτελεί μία ενέργεια
  - Συναρτήσεις (Function): Κώδικας VBA που εκτελεί υπολογισμούς και επιστρέφει μία τιμή
- Μία λειτουργική μονάδα μπορεί να περιέχει οποιοδήποτε αριθμό και συνδυασμό δηλώσεων, υπορουτινών και συναρτήσεων. Η οργάνωση του κώδικα VBA που γράφουμε είναι θέμα προσωπικής επιλογής

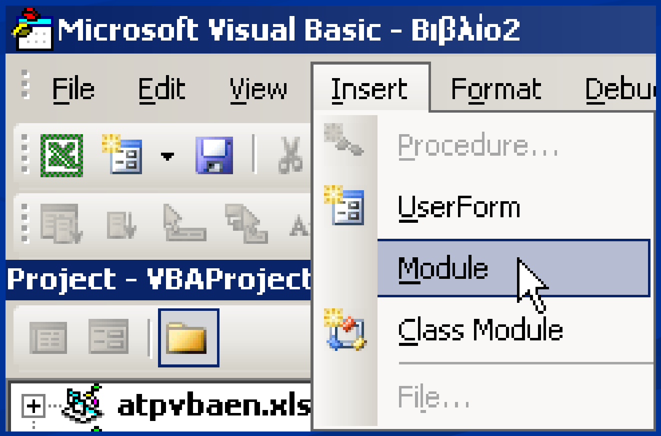
# Λειτουργικές Μονάδες (modules) Εισαγωγή

- Το VBA Project ενός βιβλίου εργασίας δεν περιέχει καμία λειτουργική μονάδα εκτός αν έχουμε καταγράψει κάποια μακροεντολή
- Εισαγωγή Λειτουργικής Μονάδας:

1ος Τρόπος

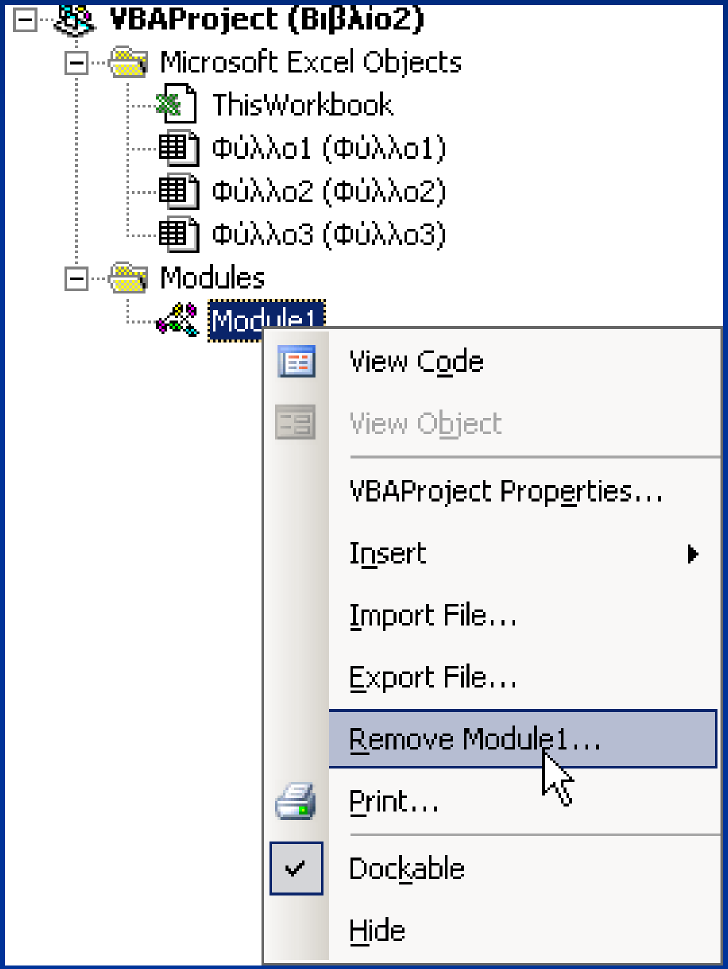



2ος Τρόπος



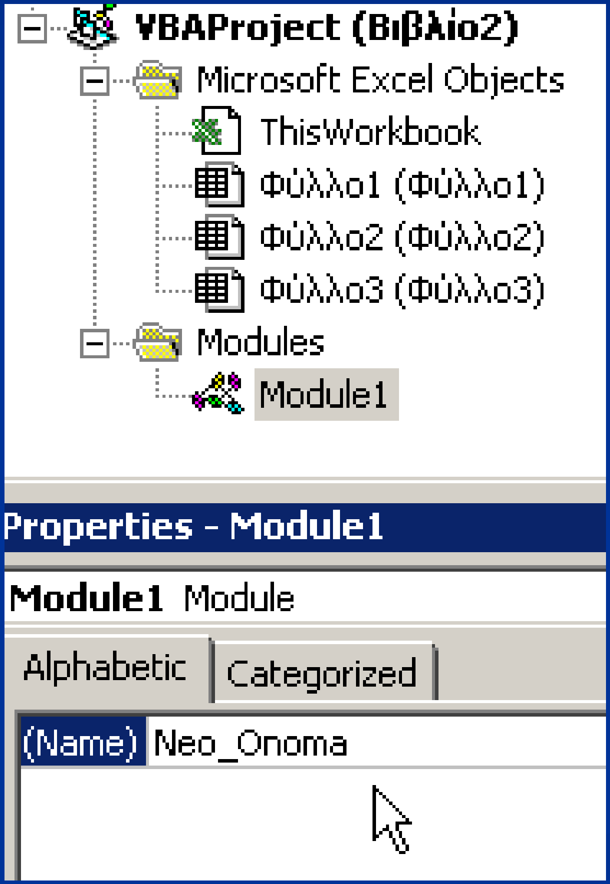
# Λειτουργικές Μονάδες (modules) Διαγραφή

- Διαγραφή λειτουργικής μονάδας:



# Λειτουργικές Μονάδες (modules) Μετονομασία

- Μετονομασία λειτουργικής μονάδας: Επιλέγουμε το module και στο παράθυρο ιδιοτήτων εισάγουμε το νέο όνομα στο πεδίο Name.



# 4. Υπορουτίνες

Εκπαιδευτικοί Στόχοι:

- Δημιουργία απλών υπορουτινών αυτοματοποίησης εργασιών μέσω της VBA

# Δήλωση Υπορουτινών

- Οι υπορουτίνες (subroutines ή sub) είναι διαδικασίες που εκτελούν μία ή περισσότερες ενέργειες.
- Οι μακροεντολές αποθηκεύονται σε μορφή υπορουτινών
- Οι υπορουτίνες μπορούν να κληθούν από άλλες διαδικασίες: υπορουτίνες ή συναρτήσεις

```
Sub Add_Cells()
    a = Cells(1, 1)
    b = Cells(1, 2)
    c = a + b
    Cells(1, 3) = c
End Sub
```

Γραμμή δήλωσης υπορουτίνας

Γραμμές κώδικα υπορουτίνας

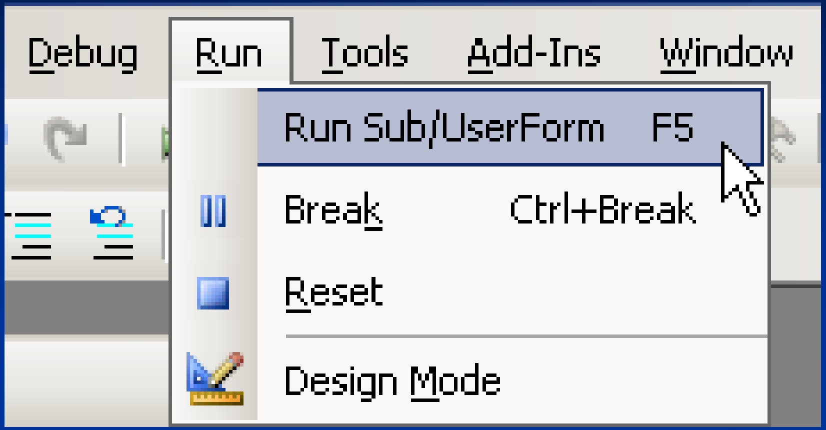
Γραμμές κώδικα υπορουτίνας

- Σε κάθε γραμμή κώδικα αντιστοιχεί μία εντολή. Αν η εντολή έχει μεγάλο μήκος και θέλουμε να την χωρίσουμε σε δύο ή περισσότερες γραμμές βάζουμε στο τέλος της κάθε γραμμής (εκτός της τελευταίας) το χαρακτήρα "_" (κάτω παύλα, underscore), π.χ.:

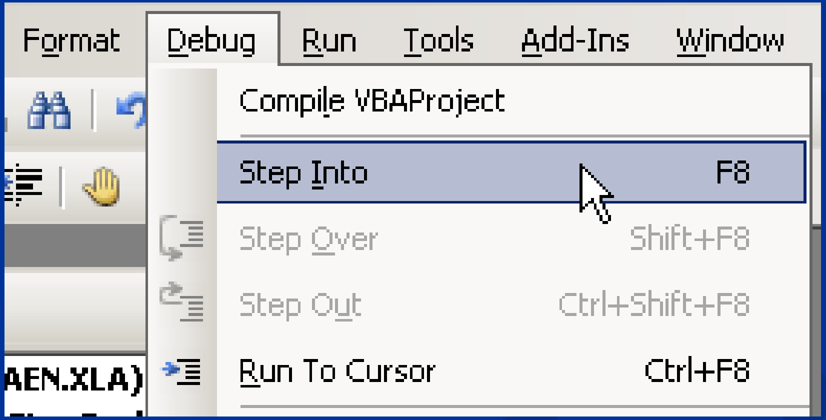
```
Application.Workbooks("Βιβλίο1.xls").Worksheets("Φύλλο1") _
        Range("A1").Value = 100
```

# Εκτέλεση Υπορουτινών

- **Ολική εκτέλεση**: κλικ με το ποντίκι οπουδήποτε εντός της υπορουτίνας και πατάμε F5 ή από το μενού [Run]→[Run Sub/UserForm]




- **Βηματική Εκτέλεση**: κλικ με το ποντίκι οπουδήποτε εντός της υπορουτίνας και πατάμε F5 ή από το μενού [Debug]→[Step Into]



# Κλήση Υπορουτινών από άλλες Υπορουτίνες

- Μπορούμε από μία υπορουτίνα να καλούμε άλλες υπορουτίνες με τους ακόλουθους τρεις τρόπους:
- ① Εισάγουμε το όνομα της υπορουτίνας και τυχόν ορίσματα που έχει χωρίζοντας τα με κόμμα
- ② Χρησιμοποιώντας την εντολή Call
- ③ Με τη χρήση της μεθόδου Run του αντικειμένου Application

①
```
Sub Master_sub()
a = 1
b = 2
Called_Sub a, b
End Sub

Sub Called_Sub(arg1, arg2)
Cells(1, 1) = arg1 + arg2
End Sub
```

②
```
Sub Master_sub()
a = 1
b = 2
Call Called_Sub(a, b)
End Sub

Sub Called_Sub(arg1, arg2)
Cells(1, 1) = arg1 + arg2
End Sub
```

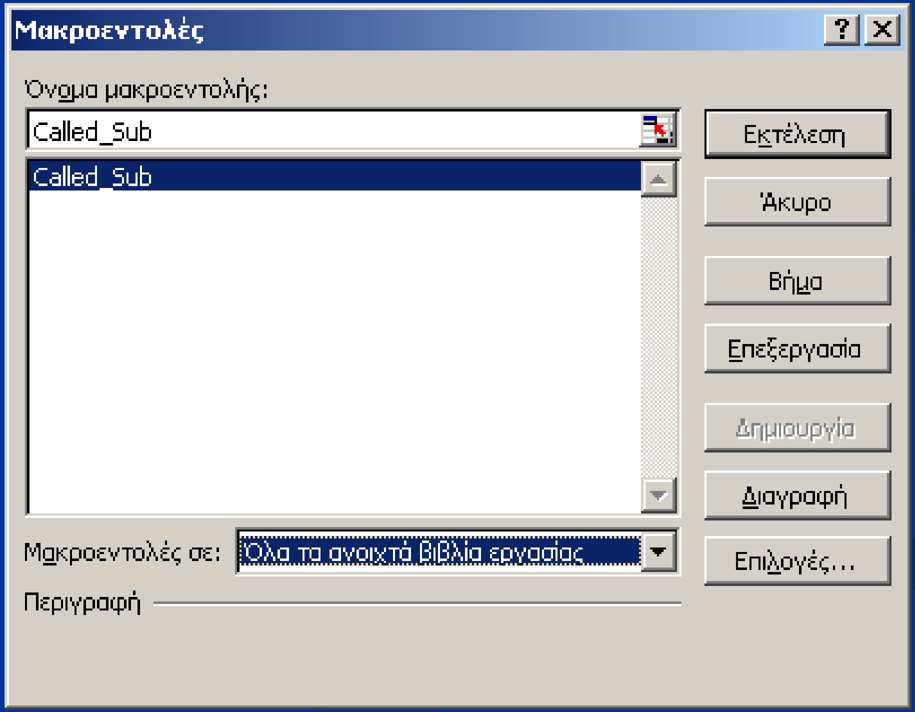
③
```
Sub Master_sub()
a = 1
b = 2
Application.Run "Called_Sub", a, b
End Sub

Sub Called_Sub(arg1, arg2)
Cells(1, 1) = arg1 + arg2
End Sub
```

# Κλήση Υπορουτινών από το περιβάλλον του Excel

- **Ο κώδικας μίας μακροεντολής περιέχεται μέσα σε μία υπορουτίνα και συνεπώς η υπορουτίνα εμφανίζεται ως μακροεντολή στο παράθυρο "Μακροεντολές" από το μενού [Εργαλεία]→[Μακροεντολές].**
- **Για να εμφανίζεται οποιαδήποτε υπορουτίνα στο παράθυρο "Μακροεντολές" δε θα πρέπει να έχει κανένα όρισμα**
- **Έτσι τροποποιούμε την προηγούμενη υπορουτίνα:**
- **Μόλις καλέσουμε την υπορουτίνα από το παράθυρο "Μακροεντολές" θα εμφανιστεί στο κελί A1 η τιμή 3.**

```
Sub Called_Sub()
arg1 = 1
arg2 = 2
Cells(1, 1) = arg1 + arg2
End Sub
```



# 5. Μεταβλητές & Τύποι Δεδομένων

Εκπαιδευτικοί Στόχοι:

- Διάκριση της σκοπιμότητας της χρήσης των μεταβλητών αλλά και της ανάγκης ύπαρξης των μεταβλητών διαφόρων τύπων
- Προσδιορισμός του είδους της μεταβλητής που χρειάζεται κάθε φορά

# Μεταβλητές

- Χώρος αποθήκευσης στη μνήμη
- Το όνομα της μεταβλητής αποτελείται από γράμματα, αριθμούς και σύμβολα. Ο πρώτος χαρακτήρας πρέπει όμως να είναι γράμμα
- Δεν επιτρέπονται οι χαρακτήρες: “.”, “ “, “#”, “$”, “%”, “&” και “!”
- Επιτρέπεται, και χρησιμοποιείται συχνά, ο χαρακτήρας “_” (underscore)

# Δήλωση Μεταβλητών

- Δηλώνουμε μία μεταβλητή ως εξής:
  Dim OnomaMetavlitis As datatype
  Όπου datatype ο τύπος δεδομένων της μεταβλητής
- Παράδειγμα:

```
Sub show_variable()
    Dim var1 As Integer, var2 As Integer
    var1 = 1
    MsgBox var1
    var2 = 2
    MsgBox var2
End Sub
```

# Μεταβλητές Παραδείγματα

- Μπορούμε να αποδώσουμε σε μία μεταβλητή μία τιμή απευθείας χωρίς προηγουμένως να τη δηλώσουμε και να ορίσουμε τον τύπο δεδομένων που αποθηκεύονται σ' αυτή, όπως στο διπλανό παράδειγμα
- Τι θα συμβεί όμως στο δεύτερο παράδειγμα;
- Προκειμένου να αποφύγουμε προβλήματα με τις μεταβλητές μπορούμε να δηλώσουμε ότι κάθε μεταβλητή πρέπει να έχει δηλωθεί πριν χρησιμοποιηθεί, εισάγοντας τη δήλωση: **Option Explicit** στην αρχή της λειτουργικής μονάδας (module)
- Τι θα συμβεί στο δεύτερο παράδειγμα μετά τη δήλωση Option Explicit;

```
Sub show_variable()
var1 = 1
MsgBox var1
var2 = 2
MsgBox var2
End Sub
```

```
Sub show_variable()
var1 = 1
MsgBox var1
var2 = 2
MsgBox var2
var1 = 3
MsgBox va1
End Sub
```

# Εύρος (scope) Μεταβλητών

- Ανάλογα με το σημείο του κώδικα και τον τρόπο δήλωσης μίας μεταβλητής, αυτή, μπορεί να είναι ενεργή:
  - Μόνο μέσα σε μία διαδικασία (υπορουτίνα)
  - Μόνο σε μία λειτουργική μονάδα (module)
  - Σε όλο το project. Δηλαδή σε όλα τα modules που περιέχει το βιβλίο εργασίας

# Εύρος (scope) Μεταβλητών

- Ανάλογα με το σημείο του κώδικα και τον τρόπο δήλωσης μίας μεταβλητής, αυτή, μπορεί να είναι ενεργή:

| **Εύρος** | **Δήλωση** | **Σημείο Δήλωσης** |
|---|---|---|
| Στη διαδικασία | Dim ή Static | Μέσα στη διαδικασία |
| Στο module | Dim ή Private | Μέσα στο module και πριν τη πρώτη διαδικασία |
| Στο project | Public | Μέσα στο module και πριν τη πρώτη διαδικασία |

# Τύποι Μεταβλητών

- **Τοπικές Μεταβλητές (Local variables)**

  Διάρκεια ζωής τους εκτείνεται μέσα στη διαδικασία. Η μεταβλητή παύει να υπάρχει όταν τελειώσει η εκτέλεση της διαδικασίας. Μπορούμε να δηλώνουμε την ίδια μεταβλητή σε διαφορετικές διαδικασίες με το ίδιο όνομα, αφού είναι μοναδική σε κάθε διαδικασία.

```
Option Explicit

Sub LifeTime()
Dim counter As Integer
counter = counter + 1
MsgBox counter
End Sub
```

- **Τοπικές Στατικές Μεταβλητές (Static variables)**

  Η διάρκεια ζωής τους εκτείνεται όσο το βιβλίο εργασίας είναι ανοικτό. Κάθε φορά που εκτελείται η διαδικασία η μεταβλητή Counter αλλάζει. Η Counter της διαδικασίας Scope1() είναι διαφορετική από αυτή της Scope2() .

```
Option Explicit

Sub Scope1()
 Static Counter As Integer
 Sales = Counter + 1
 MsgBox Counter
End Sub

Sub Scope2()
 Static Counter As Integer
 Sales = Counter + 10
 MsgBox Counter
End Sub
```

# Τύποι Μεταβλητών

- **Ιδιωτικές Μεταβλητές ορατές σε όλες τις διαδικασίες ενός module (Private variables)**

  Όταν θέλουμε μία μεταβλητή να είναι ορατή σε όλες τις διαδικασίες του συγκεκριμένου module τη δηλώνουμε πριν τη πρώτη διαδικασία.

  Η τιμή της μεταβλητής δεν αλλάζει όταν τελειώσει η εκτέλεση μίας διαδικασίας μέσα στο ίδιο module.

```
Option Explicit

Private LastName As String

Sub Exercise()
    Dim FirstName As String

    FirstName = "ΜΑΡΙΑ"
    LastName = "ΓΕΩΡΓΙΑΔΗ"

    Call Test

End Sub

Sub Test()
    MsgBox LastName
End Sub
```

## Δημόσιες Μεταβλητές (Public variables)

- Ενεργές σε όλο το project, δηλαδή σε όλα τα modules που περιέχει το βιβλίο εργασίας. Αν η μεταβλητή δηλωθεί με την έκφραση **Public** αντί για **Dim**, πριν την πρώτη διαδικασία ενός module, τότε αυτή  είναι ορατή σε όλο το project, δηλαδή σε όλα τα modules και τις υπορουτίνες του βιβλίου εργασίας.

Βιβλίο1 - Module1 (Code)

(General) | (Declarations)

```
Option Explicit

Private LastName As String
Public FullName As String

Sub Exercise()
    Dim FirstName As String

    FirstName = "ΜΑΡΙΑ"
    LastName = "ΓΕΩΡΓΙΑΔΗ"
    FullName = FirstName & " " & LastName
    Call TestPrivat
    Call TestPublic
End Sub

Sub TestPrivate()
    MsgBox LastName
End Sub
```

Βιβλίο1 - Module2 (Code)

(General) | TestPublic

```
Option Explicit

Sub TestPublic()
    MsgBox FullName
End Sub
```

# Τύποι Μεταβλητών

- **Σταθερές Μεταβλητές (Constant variables)**

  Οι σταθερές μεταβλητές είναι χρήσιμες όταν θέλουμε να διατηρούν μία σταθερή τιμή που χρησιμοποιείται σε διάφορους υπολογισμούς. Για να δηλώσουμε ότι μία μεταβλητή είναι σταθερή χρησιμοποιούμε την έκφραση **Const**. Για τις σταθερές ισχύει ότι ισχύει και με τις άλλες μεταβλητές. Μπορούν να είναι ενεργές μόνο μέσα σε μία διαδικασία, μόνο στο module που δηλώνονται ή σε ολόκληρο το project.

```
Const Pi = 3.14159265358979

Private Const Pi = 3.14159265358979

Public Const Pi = 3.14159265358979
```

# Τύποι Μεταβλητών

- **Πίνακες (Arrays)**

  Ένας πίνακας είναι μία ομάδα στοιχείων του ίδιου τύπου που χαρακτηρίζονται από ένα κοινό όνομα. Αναφερόμαστε σε ένα στοιχείο ενός πίνακα χρησιμοποιώντας το όνομα και έναν ή περισσότερους δείκτες (αριθμούς). Π.χ. ορίζουμε έναν πίνακα 12 μεταβλητών τύπου string έτσι ώστε κάθε μεταβλητή να αντιστοιχεί στο όνομα κάθε μήνα. Ονομάζουμε τον Πίνακα MonthNames. Αναφερόμαστε σε κάθε μεταβλητή του πίνακα ως: MonthNames(0), MonthNames(1) κοκ.

  Δηλαδή: MonthNames(1) = "Ιανουάριος"
  MonthNames(2) = "Φεβρουάριος"
  …
  MonthNames(12) = "Δεκέμβριος"

# Τύποι Μεταβλητών

- **<u>Δήλωση Μονοδιάστατων Πινάκων</u>**

  Dim MyArray(1 To 100) As Integer

  Dim MyArray(100) As Integer ή Dim MyArray(0 To 100) As Integer

  Στις παραπάνω δύο δηλώσεις ο πρώτος πίνακας έχει δείκτη από 1 έως 100 ενώ στη δεύτερη ο δείκτης είναι από 0 έως 100.

  Εξορισμού η αρίθμηση των δεικτών των πινάκων αρχίζει από το 0. Αν θέλουμε η αρίθμηση των δεικτών των πινάκων να αρχίζει από ένα βάζουμε πριν από κάθε διαδικασία στο module το:

  Option Base 1

# Τύποι Μεταβλητών

- **Δήλωση Πολυδιάστατων Πινάκων**

  Οι πίνακες στη VBA μπορούν να έχουν μέχρι και 60 διαστάσεις ωστόσο είναι σπάνιο να χρειαστούν περισσότερες από 3 διαστάσεις.

  Δήλωση δισδιάστατου πίνακα:

  ```
  Dim MyArray(1 To 10, 1 To 10) As Integer
  ```

  Ο παραπάνω πίνακας είναι ένας 10 X 10 Πίνακας. Για να δώσουμε τιμή στη μεταβλητή του πίνακα που βρίσκεται στην 3η γραμμή και 4η στήλη πληκτρολογούμε:

  ```
  MyArray(3,4) = 125
  ```

# Τύποι Δεδομένων

| Τύπος δεδομένων | Μέγεθος (Bytes) | Εύρος τιμών |
| --- | --- | --- |
| Byte | 1 | 0 έως 255 |
| Boolean | 2 | True ή False |
| Integer | 2 | -32.768 έως 32.767 |
| Long | 4 | -2.147.483.648 έως 2.147.483.647 |
| Single | 4 | $-3{,}402823e^{38}$ έως $-1{,}401298e^{-45}$ (αρνητικοί αριθμοί) και έως $1{,}401298e^{-45}$ έως $3{,}402823e^{38}$ (θετικοί αριθμοί) |
| Double | 8 | $-1{,}79769313486232e^{308}$ έως - $4{,}94065645841247e^{-324}$ (αρνητικοί αριθμοί) και έως $4{,}94065645841247e^{-324}$ έως $1{,}79769313486232e^{308}$ (θετικοί αριθμοί) |
| Currency | 8 | -922.337.203.685.477.5808 έως 922.337.203.685.477.5807 |
| Decimal | 14 | +/- 79.228.162.514.264.337.593.543.950.335 χωρίς δεκαδικό μέρος, και +/- 7,9228162514264337593543950335 με 28 δεκαδικά ψηφία |

# Τύποι Δεδομένων

| Τύπος δεδομένων | Μέγεθος (Bytes) | Εύρος τιμών |
|---|---|---|
| Date | 8 | 1η Ιανουαρίου 100 έως 31η Δεκεμβρίου 9999 |
| Object | 4 | Οποιαδήποτε αναφορά σε αντικείμενο του Excel |
| String (fixed length) | Μέγεθος αλφαριθμητικού | 1 μέχρι περίπου 64.500 |
| String (variable length) | 10 + μέγεθος αλφαριθμητικού | 0 μέχρι περίπου 2.000.000 |
| Variant (numbers) | 16 | Οποιαδήποτε αριθμητική τιμή εντός του εύρους τύπου double |
| Variant (characters) | 22 + μέγεθος αλφαριθμητικού | 0 μέχρι περίπου 2.000.000 |
| User defined | Εξαρτώμενο | Εξαρτώμενο από τον τύπο |

- Αν δε δηλώσουμε μία μεταβλητή, τότε αυτή χρησιμοποιεί τον τύπο δεδομένων variant.

# 6. Εργασία με Κελιά και περιοχές Κελιών

**Εκπαιδευτικοί Στόχοι:**

- Επιλογή και μορφοποίηση περιοχών φύλλου εργασίας μέσω κατάλληλων εντολών

# Το αντικείμενο Range

- Χρησιμοποιείται όταν θέλουμε να αναφερθούμε σε μία περιοχή κελιών
- Διάφορες μορφές σύνταξης
- Περιοχή κελικών: **Range("A1:C5") ή Range("A1", "C5")**
- Ένα κελί μόνο: **Range("A1")**
- Μία ή περισσότερες στήλες: **Range("A:C")**
- Μία ή περισσότερες γραμμές: **Range("4:7")**
- Για μη συνεχιζόμενες περιοχές κελιών: **Range("A1, C5, D8:E10")**
- Για περιοχή κελιών με όνομα, πχ. "Περιοχή1": **Range("Περιοχή1")**

# Βασικές Ιδιότητες του Αντικειμένου Range

- Η ιδιότητα Value του αντικειμένου Range μας επιτρέπει να ορίσουμε ή να διαβάσουμε την τιμή των κελιών μίας περιοχής.
- Για να δώσουμε στο κελί A1 την τιμή 10: **Range(“A1”).Value = 10**
- Το ίδιο αποτέλεσμα θα έχουμε και με το: **Range(“A1”)=10**
- Για να δώσουμε την ίδια τιμή σε μία περιοχή κελιών:
  **Range(“A1:B3”)=10**
- Για να διαβάσουμε την τιμή ενός κελιού:
  **MsgBox Range(“A1”).Value**
- Για να εισάγουμε έναν τύπο σε ένα ή περισσότερα κελιά:
  **Range(“A2”).Value = “=A1+5”**
  **MsgBox Range(“A2”).Value**
- Για να δούμε τη συνάρτηση που χρησιμοποιείται για ένα κελί:
  **MsgBox Range(“A2”).Formula**
- Για να εισάγουμε έναν τύπο με την ιδιότητα Fromula:
  **Range(“A2”).Formula = “=A1+5”**

```
MsgBox Range("A9").Value
MsgBox Range("A9").Address
MsgBox Range("A9").Column
MsgBox Range("A9").Row
MsgBox Range("A9").Formula
```

# Βασικές Μέθοδοι του Αντικειμένου Range

- Μέθοδοι:
  - **Select**
  - **Clear**
  - **Delete**
  - **Copy**
  - **PasteSpecial**

```
Range("A9").Select
Range("A9").Value = 5
Range("A10").Clear
Range("A11").Copy
Range("A12").PasteSpecial
Range("B12:E12").Delete
```

# Επιλογή  περιοχής κελιών

- Επιλογή περιοχής κελιών: Range(“A1:C5”).Select
- Επιλογή περιοχής συνεχόμενων  κελιών χωρίς να γνωρίζουμε τα όρια της περιοχής: **Range(“B3”).Activate ActiveCell.CurrentRegion.Select**

# Αντιγραφή, Μετακίνηση, Διαγραφή περιοχής κελιών

- Για να αντιγράψουμε μία περιοχή κελιών σε μία άλλη περιοχή (π.χ. την περιοχή A1:C5 στην περιοχή D6:F10)
  **Range(“A1:C5”).Copy Range(“D6”)**
- Παρατηρήστε ότι για την περιοχή στην οποία αντιγράφεται η αρχική περιοχή χρησιμοποιούμε μόνο το πάνω αριστερό κελί της περιοχής.
- Όμοια για να μετακινήσουμε μία περιοχή κελιών:
  **Range(“A1:C5”).Cut Range(“D6”)**
- Για να διαγράψουμε μία περιοχή κελιών:
  Range(“B3:B4”).Delete Shift:=xlShiftUp
- Η μέθοδος Delete διαγράφει την περιοχή των κελιών και όχι το περιεχόμενό τους. Η παράμετρος Shift ορίζει πως θα μετακινηθούν τα κελιά για να αντικαταστήσουν τα κελιά που διεγράφησαν. Παίρνει δύο τιμές xlShiftUp ή xlShiftToLeft
- Για να διαγράψουμε το περιεχόμενο περιοχής και όχι τα κελιά της περιοχής:
  **Range(“B3:B4”).ClearContents**
- Για να διαγράψουμε σχόλια χρησιμοποιούμε τη μέθοδο **ClearComments**
- Για να διαγράψουμε τη Μορφοποίηση χρησιμοποιούμε τη μέθοδο: **ClearFormats**
- Για να διαγράψουμε ταυτόχρονα Περιεχόμενο, Μορφοποίηση και Σχόλια: **Clear**

# Μορφοποίηση Περιοχής Κελιών

- Η μορφοποίηση της γραμματοσειράς σε μία περιοχή κελιών ελέγχεται από το αντικείμενο Font το οποίο έχει μία σειρά ιδιοτήτων: Bold, Italic, Size, Color κ.α., πχ.:
  Range(“A1:A2”).Font.Bold = True
  Range(“A1:A2”).Font.Size = 12
  Range(“A1:A2”).Font.Color = RGB(0,255,0)
  Range(“A1:A2”).Font.Name = “Arial Black”
- Τα αντικείμενα Collumns και Rows που ανήκουν στο Range έχουν τις ιδιότητες ColumnWidth και RowHeight:
  Range(“B3:C4”).Columns.ColumnWidth = 20
- Range(“B3:C4”).Rows.RowHeight = 20
- Η μέθοδος **AutoFit** δίνει τη δυνατότητα να ορίσουμε το πλάτος ή το ύψος έτσι ώστε να προσαρμόζονται αυτόματα στο περιεχόμενο των κελιών της περιοχής:
  Range(“B3:C4”).Columns.AutoFit

# Η ιδιότητα Cells

- Αν θέλουμε να εργαστούμε με ένα κελί χρησιμοποιούμε την ιδιότητα Cells που ανήκει στα αντικείμενα Application, ActiveSheet και Range, π.χ για να επιλέξουμε το κελί B3 γράφουμε: Cells(2,3)
- Ο πρώτος αριθμός δηλώνει τη γραμμή (1 έως 65536) και ο δεύτερος τη στήλη (1 έως 256
- Το πλεονέκτημα της Cell έναντι της Range είναι ότι μπορούμε να χρησιμοποιήσουμε μεταβλητές για να δίνουμε τις τιμές που θέλουμε στις παραμέτρους.
- Για να επιλέξουμε όλα τα κελιά του ενεργού φύλλου γράφουμε: Application.Cells ή ActiveSheet.Cells

```
Sub Set_Cells_Values()
For N = 1 To 5
Cells(N + 5, 1) = 3 * N + N ^ 2
Next N
End Sub
```

# 7. Εισαγωγή στο Excel Object Model

**Εκπαιδευτικοί Στόχοι:**

- Διάκριση των βασικών αντικειμένων του Excel και χρήση των βασικών ιδιοτήτων τους

# Εισαγωγή στο Excel Object Model

- Η VBA χρησιμοποιεί αντικείμενα για να ελέγχει το Excel.
- Κάθε αντικείμενο αντιπροσωπεύει ένα στοιχείο του Excel
- Κάθε αντικείμενο είτε περιέχει άλλα αντικείμενα, είτε περιέχεται σε άλλα αντικείμενα
- Τα αντικείμενα στο Excel τοποθετούνται σε μία ιεραρχική δομή που είναι το Excel Object Model
- Το Excel Object Model περιγράφει τις σχέσεις μεταξύ των αντικειμένων
- Τα πιο χρησιμοποιούμενα αντικείμενα είναι:
  - Application
  - Workbook
  - Worksheet
  - Range
  - Form
  - Chart
  - Dialog

# Εισαγωγή στο Excel Object Model

- Ένα αντικείμενο που περιέχει περισσότερα από ένα όμοια αντικείμενα είναι μία συλλογή (collection)
- Η συλλογή Worksheets αποτελείται από τα αντικείμενα Worksheet, που είναι τα φύλλα ενός βιβλίου εργασίας
- Για κάθε αντικείμενο υπάρχει μία κλάση (class) που είναι ένα πρότυπο που ορίζει το αντικείμενο, τις ιδιότητες, τις μεθόδους και τα γεγονότα που το απαρτίζουν
- Η γραμμή κώδικα **Workbooks("Βιβλίο2").Activate** επιλέγει το αντικείμενο "Βιβλίο2" από τη συλλογή αντικειμένων Workbooks
- Ο κώδικας για να αναφερθούμε σε μία ιδιότητα η μέθοδο ενός αντικειμένου είναι **object.property** ή **object.method**
- Μπορεί να χρησιμοποιήσουμε περισσότερα αντικείμενα, στην ιεραρχία, και να καλέσουμε μία ιδιότητα ενός αντικειμένου:
ActiveSheet.Range("A1").Font.Bold = True
object.object.object.property = True
- Ο καλύτερος τρόπος για να διαβάσουμε την παραπάνω γραμμή κώδικα είναι από δεξιά προς τα αριστερά

# ΠΑΡΑΔΕΙΓΜΑ

1. Αποθηκεύστε ότι έχετε κάνει μέχρι τώρα
2. Ανοίξτε το Excel και δημιουργείστε 3 βιβλία εργασίας (Βιβλίο1, Βιβλίο2 και Βιβλίο3)
3. Επιλέξτε/Ενεργοποιείστε το Βιβλίο1
4. Ανοίξτε τον επεξεργαστή VBA με Alt+F11
5. Δημιουργήστε ένα νέο module και μία υπορουτίνα η οποία θα γράφει το κείμενο "Κατάλογος Ονομάτων" στο κελί A2 του φύλλου εργασίας "Φύλλο2" του βιβλίου εργασίας "Βιβλίο2"

1ος τρόπος

```
Workbooks("Βιβλίο2").Activate
Worksheets("Φύλλο2").Activate
Range("A2").Value = "Κατάλογος Ονομάτων"
```

2ος τρόπος

```
Workbooks("Βιβλίο2").Worksheets("Φύλλο2").Range("A2").Value = "Κατάλογος Ονομάτων"
```

# Το Αντικείμενο Application

- Είναι το πρώτο αντικείμενο στην ιεραρχία που για το Excel είναι το ίδιο το Excel
- Περιέχει όλες τις ιδιότητες και τις μεθόδους που είναι απαραίτητες για να προγραμματίσουμε το ίδιο το Excel καθώς και όλα τα υπόλοιπα αντικείμενα
- Επειδή είναι το πρώτο στην ιεραρχία είναι μοναδικό και σπάνια χρειάζεται να το καλούμε
- Παραδείγματα χρήσης του αντικειμένου Application:
  - Application.Width = 600
  - Application.Height = 450
  - Application.DisplayFormulaBar = False

# Παράδειγμα

- Τι κάνει ο παρακάτω κώδικας;

```
Sub ShowAppSize()
' Μεγιστοποίηση Παραθύρου Excel
Application.WindowState = xlMaximized

' Μεταβλητές Πλάτους και μήκους παραθύρου Excel
appWidth = Application.Width
appHeight = Application.Height

' Παράθυρο μηνύματος με το Πλάτος και Μήκος
Msg = "Το μέγεθος του παραθύρου του Excel είναι: "
Msg = Msg & appWidth & " X " & appHeight
MsgBox Msg
End Sub
```

# Η συλλογή Workbooks

- Η συλλογή **Workbooks** αποτελείται από αντικείμενα **Workbook**
- Με χρήση των μεθόδων που έχει μπορούμε να δημιουργήσουμε ένα νέο βιβλίο εργασίας, να κλείσουμε ένα ανοικτό βιβλίο εργασίας ή να ανοίξουμε ένα αποθηκευμένο βιβλίο εργασίας.
- Για να δημιουργήσουμε ένα νέο βιβλίο εργασίας χρησιμοποιούμε τη μέθοδο **Add**:
  **Workbooks.Add**
- Για να κλείσουμε ένα βιβλίο εργασίας γράφουμε:
  **Workbooks("Βιβλίο1").Close**
- Για να αποφύγουμε το μήνυμα αν θέλουμε να αποθηκεύσουμε τις αλλαγές γράφουμε:
  Workbooks("Βιβλίο1").Close SaveChanges:=True
- Αν δώσουμε στην παράμετρο SaveChanges την τιμή False, το Excel κλείνει το βιβλίο εργασίας χωρίς να αποθηκεύσει τις αλλαγές.

# Ασκήσεις

1. Τι νομίζετε πως μπορεί να κάνει η διπλανή υπορουτίνα; Εκτελέστε τον κώδικα για να ελέγξετε την απάντησή σας

```
Sub AddWorkbooks()
    Dim I As Integer
    For I = 1 To 3
        Workbooks.Add
    Next I
End Sub
```

2. Τι κάνει η διπλανή υπορουτίνα; Εκτελέστε τον κώδικα για να ελέγξετε την απάντησή σας.

```
Sub CloseLastWorkbook()
    Dim numWorkbooks As Integer
    numWorkbooks = Workbooks.Count
    Workbooks(numWorkbooks).Activate
    Workbooks(numWorkbooks).Close (True)
    ' ή Workbooks(numWorkbooks).Close SaveChanges:=True
End Sub
```

# Το αντικείμενο Workbook - Ιδιότητες

- Το αντικείμενο **Workbook** αναφέρεται σε ένα βιβλίο εργασίας και έχει περισσότερες ιδιότητες και μεθόδους από τη συλλογή Workbooks.

- Οι πιο χρήσιμες ιδιότητες είναι:

| | |
|---|---|
| **ActiveChart** | Επιστρέφει το ενεργό γράφημα |
| **ActiveSheet** | Επιστρέφει το ενεργό φύλλο εργασίας |
| **FullName** | Επιστρέφει το όνομα και τη διαδρομή που είναι αποθηκευμένο το βιβλίο εργασίας |
| **Name** | Επιστρέφει το όνομα του βιβλίου εργασίας |
| **Password** | Αποδίδει ή επιστρέφει το password που θέλουμε να δώσουμε ή έχει ήδη το βιβλίο |
| **Path** | Επιστρέφει τη διαδρομή που είναι αποθηκευμένο το βιβλίο |
| **Sheets** | Επιστρέφει μία συλλογή αντικειμένων Worksheets από τα φύλλα που περιέχει το βιβλίο |
| **Worksheets** | το ίδιο |

# Το αντικείμενο Workbook - Μέθοδοι

- Οι πιο χρήσιμες μέθοδοι είναι:

| | |
|---|---|
| **Activate** | Ενεργοποιεί το βιβλίο |
| **Close** | Κλείνει το βιβλίο |
| **PrintOut** | Εκτυπώνει το βιβλίο |
| **PrintPreview** | Δημιουργεί προεπισκόπηση εκτύπωσης του βιβλίου |
| **Protect** | Ορίζει αν το βιβλίο θα προστατεύεται από αλλαγές ή όχι |
| **Save** | Αποθηκεύει το βιβλίο με το ίδιο όνομα |
| **SaveAs** | Αποθηκεύει το βιβλίο με νέο όνομα |
| **SaveCopyAs** | Αποθηκεύει ένα αντίγραφο του βιβλίου |

# Το αντικείμενο Workbook

- Για να αναφερθούμε στο βιβλίο εργασίας, στο οποίο γράφουμε κώδικα VBA, χρησιμοποιούμε την ιδιότητα **ThisWorkbook** του αντικειμένου **Application**
- Η ιδιότητα αυτή επιστρέφει ένα αντικείμενο **Workbook** που αντιστοιχεί στο βιβλίο που περιέχει τον κώδικα που εκτελούμε.
- Αν π.χ. έχουμε ανοικτά τα βιβλία Βιβλίο1.xls και Βιβλίο2.xls, και σε μία υπορουτίνα που βρίσκεται στο Βιβλίο1.xls γράφουμε:

  ```
  MsgBox ThisWorkbook.Name
  ```

  όταν εκτελεστεί θα εμφανιστεί ένα πλαίσιο διαλόγου με το μήνυμα "Βιβλίο1"

# Παράδειγμα

- Αποθήκευση βιβλίου εργασίας με νέο όνομα στο φάκελο C:\VBA Files

```
Sub Save_Workbook()
    ThisWorkbook.SaveAs Filename:= _
        "C:\VBA Files\Saved_VBA_Example.xls"
End Sub
```

# Το αντικείμενο Workbook - Συμβάντα

- Τα συμβάντα μας επιτρέπουν να εκτελούμε, εντολές αμέσως μετά την ενεργοποίησή τους
- Τα πιο χρήσιμα συμβάντα είναι:

| | |
|---|---|
| **Activate** | Εκτελεί τον κώδικα όταν ενεργοποιείται το βιβλίο |
| **BeforeClose** | Εκτελεί τον κώδικα πριν κλείσει το βιβλίο |
| **BeforePrint** | Εκτελεί τον κώδικα πριν εκτυπωθεί το βιβλίο |
| **BeforeSave** | Εκτελεί τον κώδικα πριν αποθηκευτεί το βιβλίο |
| **Deactivate** | Εκτελεί τον κώδικα όταν απενεργοποιείται το βιβλίο |
| **NewSheet** | Εκτελεί τον κώδικα όταν εισαχθεί νέο φύλλο εργασίας |
| **Open** | Εκτελεί τον κώδικα μόλις ανοίξει το βιβλίο |
| **SheetActivate** | Εκτελεί τον κώδικα μόλις ενεργοποιηθεί οποιοδήποτε φύλλο του βιβλίου |
| **SheetChange** | Εκτελεί τον κώδικα μόλις γίνει κάποια αλλαγή σε οποιοδήποτε φύλλο του βιβλίου |
| **SheetDeactivate** | Εκτελεί τον κώδικα μόλις απενεργοποιηθεί το ενεργό φύλλο του βιβλίου |

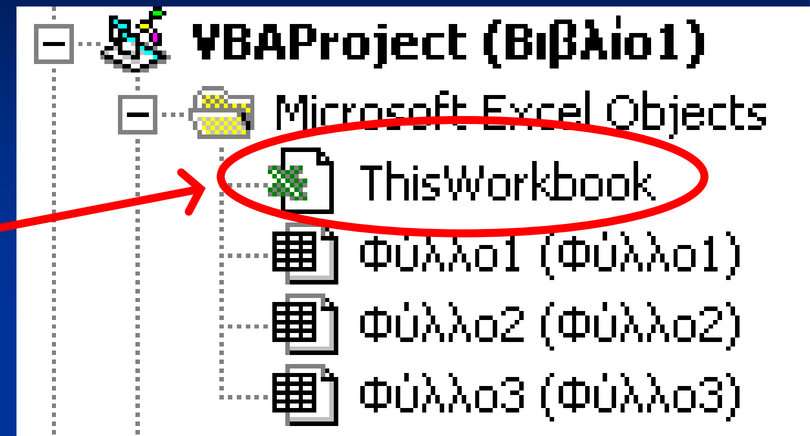
# Το αντικείμενο Workbook - Συμβάντα

- Για να εκτελεστούν οι εντολές κώδικα που επιθυμούμε με την ενεργοποίηση ενός συμβάντος κάνουμε τα ακόλουθα:

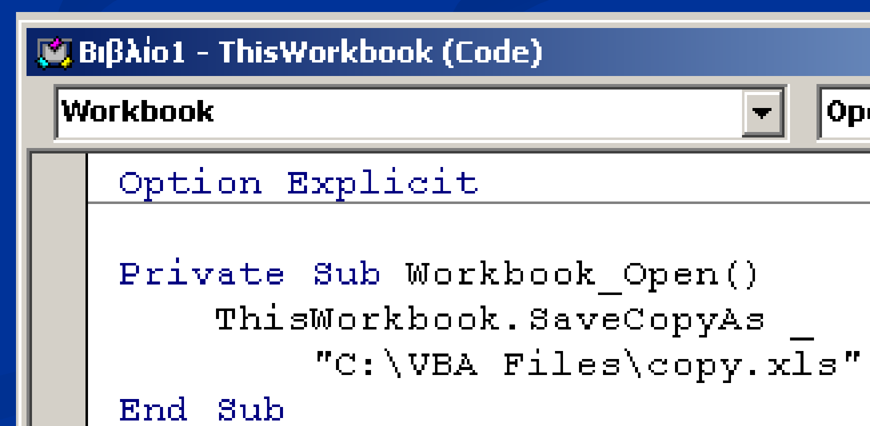
1. Ενεργοποιούμε το παράθυρο κώδικα του αντικειμένου ThisWorkbook
2. Ονομάζουμε την υπορουτίνα που θα εκτελέσει τις εντολές με τον ακόλουθο τρόπο:

   **Workbook_Συμβάν()**

   **π.χ. Workbook_Open()**




- **ΠΑΡΑΔΕΙΓΜΑ:** Δημιουργία αντιγράφου ασφαλείας με όνομα copy.xls στη διαδρομή "C:\VBA Files", κάθε φορά που ανοίγουμε το βιβλίο εργασίας



# Οι συλλογές Worksheets και Sheets

- Η συλλογή **Worksheets** αποτελείται από όμοια αντικείμενα **Worksheet** και περιέχει αναφορά σε κάθε φύλλο εργασίας του βιβλίου
- Η συλλογή **Sheets** αποτελείται από όμοια αντικείμενα **Worksheet** και **Chart** και περιέχει αναφορά σε κάθε φύλλο εργασίας καθώς και στα φύλλα γραφημάτων ενός βιβλίου
- **Οι σημαντικότερες ιδιότητές τους είναι**:
  - **Count** Επιστρέφει το πλήθος των φύλλων στο βιβλίο (για τη συλλογή Worksheet μόνο το πλήθος των φύλλων εργασίας, ενώ για τη Sheets το πλήθος των φύλλων και των φύλλων γραφήματος)
  - **Visible** Ορίζει αν τα φύλλα είναι ορατά ή όχι

# Παράδειγμα

- Εύρεση πλήθους φύλλων εργασίας που περιέχει το βιβλίο και απόκρυψη του τελευταίου φύλλου:

```
Sub hide_last_sheet()
    Dim sheet_count As Integer
    sheet_count = ThisWorkbook.Worksheets.Count
    ThisWorkbook.Worksheets(sheet_count).Visible = False
End Sub
```

# Οι συλλογές Worksheets και Sheets

- Οι χρησιμότερες μέθοδοι των συλλογών Worksheets και Sheets είναι:
  - **Add** Προσθέτει ένα νέο φύλλο εργασίας
  - **Copy** Αντιγράφει ένα φύλλο της συλλογής
  - **Delete** Διαγράφει ένα φύλλο της συλλογής
  - **Move** Μετακινεί ένα φύλλο της συλλογής
  - **PrintOut** Εκτυπώνει ένα φύλλο της συλλογής
  - **PrintPreview** Δημιουργεί προεπισκόπηση εκτύπωσης του φύλλου της συλλογής
  - **Select** Επιλέγει ένα φύλλο από τη συλλογή

# Παράδειγμα

- Προσθέτουμε δύο νέα φύλλα εργασίας (**Count:=2**) πριν το τελευταίο φύλλο (**Before:=Worksheets(Worksheets.Count)**)

```
Sub Add_sheets()
    ActiveWorkbook.Worksheets.Add _
        Before:=Worksheets(Worksheets.Count), _
        Count:=2
End Sub
```

- Αν θέλαμε να προσθέσουμε τα φύλλα μετά από κάποιο φύλλο, τότε αντί για την παράμετρο **Before** θα χρησιμοποιούσαμε την **After**
- Επίσης χρησιμοποιήσαμε την ιδιότητα ActiveWorkbook του αντικειμένου Application για να αναφερθούμε στο ενεργό βιβλίο εργασίας, το οποίο μπορεί να είναι το ίδιο ή διαφορετικό από αυτό στο οποίο βρίσκεται ο κώδικας που εκτελούμε

# Παράδειγμα

- Αντιγραφή φύλλου

```
Sub Copy_sheets()
    ActiveWorkbook.Worksheets(1).Copy _
        After:=Worksheets(Worksheets.Count)
End Sub
```

- Διαγραφή φύλλου

```
Sub Delete_sheets()
    ActiveWorkbook.Worksheets(1).Delete
End Sub
```

# Το αντικείμενο Worksheet - Ιδιότητες

- Το αντικείμενο Worksheet αναφέρεται σε ένα φύλλο εργασίας
- Οι πιο χρήσιμες ιδιότητές του είναι:
  - **Index** Επιστρέφει τον δείκτη που αντιστοιχει στο φύλλο εργασίας
  - **Name** Ορίζει ή επιστρέφει το όνομα του φύλλου
  - **Names** Επιστρέφει μία συλλογή από αντικείμενα Range στα οποία έχουν αποδοθεί ονόματα
  - **Next** Επιστρέφει το επόμενο φύλλο ως αντικείμενο Worksheet
  - **Previous** Επιστρέφει το προηγούμενο φύλλο ως αντικείμενο Worksheet
  - **Visible** Ορίζει αν το φύλλο είναι ορατό

# Παράδειγμα

- Ανατρέχουμε σε όλα τα φύλλα εργασίας του ενεργού βιβλίου και εμφανίζουμε το όνομα και το δείκτη που αντιστοιχεί σε κάθε φύλλο εργασίας

```
Sub Display_sheet_name_index()
    Dim wrksht As Worksheet
    For Each wrksht In ActiveWorkbook.Worksheets
        MsgBox "Το φύλλο που ονομάζεται " & wrksht.Name & _
        " έχει δείκτη " & wrksht.Index, , _
        "Ποια φύλλα έχει το βιβλίο;"
    Next wrksht
End Sub
```

# Παράδειγμα 2

- Θα γράψουμε μία υπορουτίνα συμβάντος για το Φύλλο2 του ενεργού βιβλίου εργασίας η οποία θα ενεργοποιείται όταν ο χρήστης ενεργοποιήσει το Φύλλο2
- Η υπορουτίνα θα πρέπει να γραφεί στο παράθυρο κώδικα του Φύλλου 2

```
Private Sub Worksheet_Activate()
    Dim msgOutput As String
    msgOutput = "Αυτό το Φύλλο είναι το " & _
        Workheets(2).Name
    MsgBox (msgOutput)
End Sub
```

# Το αντικείμενο Worksheet - Μέθοδοι

- Οι πιο χρήσιμες μέθοδοι του αντικειμένου Worksheet είναι:
  - **Activate** Ενεργοποιεί το φύλλο εργασίας
  - **Copy** Αντιγράφει το φύλλο εργασίας
  - **Delete** Διαγράφει το φύλλο εργασίας
  - **Move** Μετακινεί το φύλλο εργασίας
  - **Paste** Επικολλάει τα περιεχόμενα του Πρόχειρου στο φύλλο εργασίας
  - **PasteSpecial** Επικολλάει τα περιεχόμενα του Προχείρου στο φύλλο εργασίας
  - **PivotTable** Επιστρέφει μία Αναφορά Συγκεντρωτικού Πίνακα (Pivot Table) μία συλλογή από αναφορές
  - **PivotTableWizard** Δημιουργεί Αναφορά Συγκεντρωτικού Πίνακα
  - **PrintOut** Εκτυπώνει το φύλλο εργασίας
  - **PrintPreview** Δημιουργεί προεπισκόπηση εκτύπωσης του φύλλου
  - **Protect** Προστατεύει το φύλλο εργασίας από αλλαγές
  - **ResetAllPageBrakes** Απαλείφει τυχόν ορισμένες περιοχές εκτύπωσης
  - **SaveAs** Αποθηκεύει το φύλλο εργασίας σε ένα νέο βιβλίο
  - **Select** Επιλέγει το φύλλο εργασίας
  - **Unprotect** Ακυρώνει τυχόν προστασία του φύλλου εργασίας από αλλαγές

# Το αντικείμενο Worksheet - Συμβάντα

- Τα συμβάντα του αντικειμένου Worksheet είναι:
  - **Activate** Εκτελεί τον κώδικα όταν ενεργοποιείται το φύλλο εργασίας
  - **BeforeDoubleClick** Εκτελεί τον κώδικα αμέσως πριν από το αποτέλεσμα που θα προκαλέσει το διπλό κλικ του ποντικιού στο φύλλο εργασίας
  - **BeforeRightClick** Εκτελεί τον κώδικα αμέσως πριν από το αποτέλεσμα που θα προκαλέσει το δεξί κλικ του ποντικιού στο φύλλο εργασίας
  - **Calculate** Εκτελεί τον κώδικα όταν υπολογίζονται οι τιμές των κελιών του φύλλου εργασίας
  - **Change** Εκτελεί τον κώδικα όταν αλλάζει η τιμή ενός ή περισσοτέρων κελιών
  - **Deactivate** Εκτελεί τον κώδικά όταν απενεργοποιείται το φύλλο εργασίας
  - **FollowHyperlink** Εκτελεί τον κώδικα όταν κάνουμε κλικ σε μια υπερσύνδεση που βρίσκεται στο φύλλο εργασίας
  - **PivotTabelUpdate** Εκτελεί τον κώδικα όταν ανανεώνονται τα περιεχόμενα μίας Αναφοράς Συγκεντρωτικού Πίνακα που βρίσκεται στο φύλλο εργασίας
  - **SelectionChange** Εκτελεί τον κώδικα όταν επιλέγουμε μία άλλη περιοχή κελιών του φύλλου εργασίας

# 8. Συναρτήσεις – Συναρτήσεις οριζόμενες από το χρήστη

**Εκπαιδευτικοί Στόχοι:**

- Ανάλυση των ωφελειών από την εφαρμογή της λογικής "Διαίρει και Βασίλευε" στη δημιουργία εφαρμογών
- Περιγραφή των διαφορών που παρουσιάζουν οι συναρτήσεις σε σχέση με τις υπορουτίνες
- Δημιουργία απλών συναρτήσεων και κλήση τους με διάφορους τρόπους

# Συναρτήσεις (Functions)

- Οι συναρτήσεις είναι διαδικασίες και περιέχουν κώδικα, όπως και οι υπορουτίνες
- Οι συναρτήσεις, αντίθετα από τις υπορουτίνες, επιστρέφουν μία τιμή.
- Περιέχουν ορίσματα (arguments) που τα χρησιμοποιούν για τους υπολογισμούς που εκτελούν στο εσωτερικό τους
- Μπορούν να καλούνται από άλλες διαδικασίες ή άλλες συναρτήσεις
- Μπορούμε να δημιουργήσουμε δικές μας συναρτήσεις που καλούνται συναρτήσεις χρήστη (UDF, User Defined Functions)

# Έτοιμες Συναρτήσεις του Excel

| Τύπος Συναρτήσεων | Ενδεικτικές Συναρτήσεις |
|---|---|
| Κειμένου | Left, Right, Mid, Find, Len, … |
| Αριθμητικές / Μαθηματικές | Sum, Sumif, Countif, Subtotal, Sumproduct, Int, Round, Count, CountA |
| Στατιστικές | Average, Max, Min, Large, Small |
| Λογικές | IF, And, Or, Not |
| Πληροφορίας | IsError, IsNumber, IsText, IsBlank |
| Ημερομηνίας & Ώρας | Date, Day, Month, Year, Today(), Now(), Weekday |
| Βάσεων Δεδομένων | DSum, DAverage, DMin, DMax, DCount, DCountA |
| Αναζήτησης και Αναφοράς | Vlookup, Hlookup, Index, Offset |
| Οικονομικές | Pmt, Rate, Npv, FV, PV, Ipmt, Ppmt, |
| | |

# Έτοιμες Συναρτήσεις της VBA

| Τύπος Συναρτήσεων | Ενδεικτικές Συναρτήσεις |
|---|---|
| Κειμένου | Asc, Chr, Format, InStr, Str |
| Αριθμητικές / Μαθηματικές | Val, Format, Rnd |
| Λογικές | IF-THEN-ELSE, Case, IIF |
| Πληροφορίας | IsNull, IsDate, IsNumeric |
| Ημερομηνίας & Ώρας | Date, DatePart, Format |
| Αλλαγής Τύπου Δεδομένων | CBool, CDate, CDbl, CInt, CLng, CSt, CVar |
| Αρχείων / Καταλόγων | ChDir, FileDateTime, FileLen, MkDir |
| | |

# Δήλωση μίας Συνάρτησης

Οι συναρτήσεις όπως και οι υπορουτίνες, δηλώνονται με ένα συγκεκριμένο τρόπο. Η πλήρης σύνταξη έχει την παρακάτω μορφή:

```
[Public, Private] [Static] Function function_name([argument_list]) [As data_type]
        [VBA_instructions]
        [function_name = expression]
        [Exit Function]
        [VBA_instructions]
        [function_name = expression]
End Function
```

Οτιδήποτε είναι μέσα σε [ ] είναι προαιρετικό. Τα υπόλοιπα είναι υποχρεωτικά.

# Δήλωση Συνάρτησης

| Στοιχείο | Επεξήγηση |
|---|---|
| **Public** | Η συνάρτηση μπορεί να κληθεί από όλες τις διαδικασίες και από οποιοδήποτε module του VBA Project |
| **Private** | Η συνάρτηση μπορεί να κληθεί από όλες τις διαδικασίες του τρέχοντος module |
| **Static** | Οι τιμές των μεταβλητών της συνάρτησης διατηρούνται αναλλοίωτες μεταξύ διαδοχικών κλήσεων της συνάρτησης |
| **Function** | Έκφραση που δηλώνει την έναρξη της συνάρτησης |
| **funtion_name** | Το όνομα της συνάρτησης |
| **argument_list** | Ένα ή περισσότερα ορίσματα (χωρίζονται με κόμμα) που περνάνε στη συνάρτηση |
| **data_type** | Ο τύπος δεδομένων που επιστρέφει η συνάρτηση |
| **VBA_instructions** | Μία ή περισσότερες εντολές της VBA |
| **Exit Function** | Μία δήλωση που υποχρεώνει τη VBA να σταματήσει την εκτέλεση του κώδικα σε αυτό το σημείο και να επιστρέψει στη διαδικασία που κάλεσε τη συνάρτηση |
| **End Function** | Δηλώνει το τέλος της συνάρτησης. |

# Παράδειγμα: Δημιουργία Συνάρτησης

Οι συναρτήσεις που ακολουθούν μετατρέπουν το περιεχόμενο ενός κελιού (αριθμός) από Δραχμές σε Ευρώ και το αντίστροφο:

```
Option Explicit

Function EUR(DRX As Double) As Double
    EUR = DRX / 340.75
End Function

Function DRX(EUR As Double) As Double
    DRX = EUR * 340.75
End Function
```

# Εκτέλεση Συνάρτησης

- Μία υπορουτίνα μπορεί να εκτελεστεί πατώντας το κουμπί F5 από το μενού Run>Run Sub/UserForm

- Μία συνάρτηση δεν μπορεί να εκτελεστεί από μόνη της επειδή πρέπει να επιστρέψει μία τιμή. Επομένως θα πρέπει να κληθεί είτε από μία άλλη διαδικασία (υπορουτίνα ή συνάρτηση) ή μέσα από το περιβάλλον του Excel ως **Συνάρτηση Χρήστη.**

# Κλήση Συνάρτησης από Διαδικασία

- Για να καλέσουμε μία συνάρτηση μέσα από μία άλλη διαδικασία θα πρέπει να έχουμε εξασφαλίσει ότι η επιστρεφόμενη τιμή από τη συνάρτηση θα αποδοθεί σε κάποια μεταβλητή ή μία έκφραση η οποία μπορεί να πάρει τιμές.

```
Function DRX(EUR As Double) As Double
    DRX = EUR * 340.75
End Function
```

```
Sub Call_function()
    Dim draxmes As Double
    Dim evro As Double

    draxmes = DRX(Cells(1, 1))
    MsgBox draxmes

End Sub
```

# Κλήση Συνάρτησης από το Φύλλο Εργασίας

- Οι συναρτήσεις που δημιουργούμε στη VBA μπορούν να χρησιμοποιηθούν όπως οι γνωστές συναρτήσεις του Excel.

- Μπορούμε σε ένα κελί να πληκτρολογήσουμε το όνομα της συνάρτησης και να δώσουμε τα ανάλογα ορίσματα.

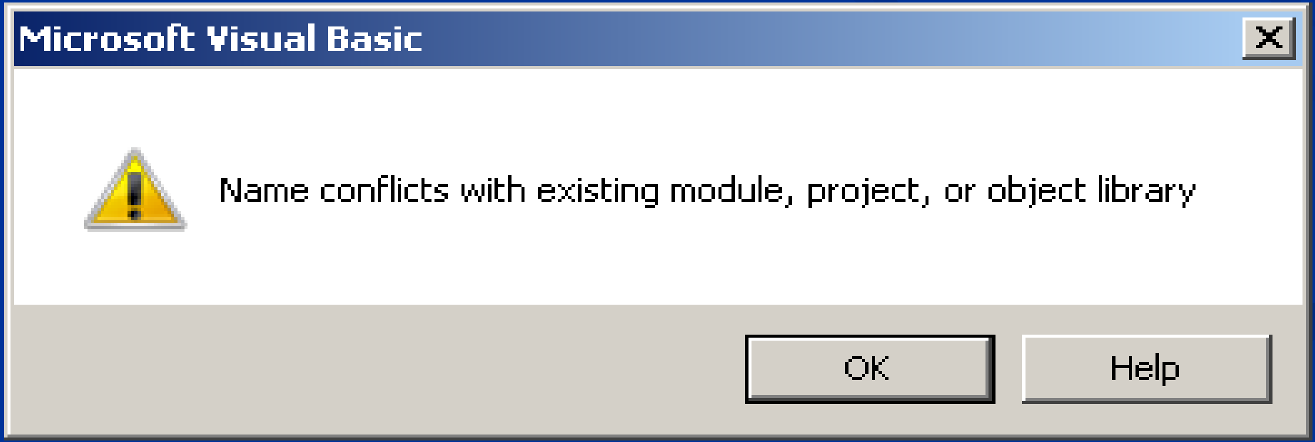
- Η συνάρτηση εμφανίζεται επίσης στο παράθυρο Εισαγωγής Συναρτήσεων στην κατηγορία **Συναρτήσεις Χρήστη.**

- Μπορούμε να αποθηκεύσουμε δικές μας συναρτήσεις σε ένα βιβλίο εργασίας και να τις καλούμε από οποιοδήποτε άλλο βιβλίο εργασίας.

## Κλήση Αποθηκευμένων Συναρτήσεων από Άλλα Φύλλα Εργασίας

1. Κατασκευάζουμε τις συναρτήσεις μας σε ένα module ενός βιβλίου εργασίας.
2. Αποθηκεύουμε το βιβλίο εργασίας με ένα συγκεκριμένο όνομα σε ένα συγκεκριμένο φάκελο του υπολογιστή που γνωρίζουμε ότι δεν θα αλλάξει.
3. Μετονομάζουμε το Project* αυτού του βιβλίου εργασίας από το εξορισμού VBAProject π.χ. σε MyFunctions.
    1. Πάμε στο Tools>VBAProject Properties
    2. Στην καρτέλα General αλλάζουμε το όνομα του project σε MyFunctions
    3. Πατάμε το πλήκτρο <OK>

***ΠΡΟΣΟΧΗ** Η διαδικασία αυτή πρέπει να γίνει αλλιώς θα εμφανιστεί το μήνυμα όταν θα πάμε να κάνουμε αναφορά στο βιβλίο με τις συναρτήσεις ( δες επόμενο βήμα)



# Κλήση Αποθηκευμένων Συναρτήσεων από Άλλα Φύλλα Εργασίας

4. Ανοίγουμε ένα άλλο βιβλίο εργασίας
5. Στον VBA Editor πάμε στο Tools>References
6. Πατάμε το κουμπί <Browse>
7. Στο πλαίσιο **Files of type** επιλέγουμε **Microsoft Office Excel Files**
8. Πάμε στο φάκελο όπου έχουμε αποθηκεύσει το βιβλίο εργασίας με τις δικές μας συναρτήσεις το επιλέγουμε και πατάμε το κουμπί <Open>

# Κλήση ένθετης Συνάρτησης του Excel από τη VBA

- Σχεδόν όλες οι ένθετες συναρτήσεις του Excel είναι δυνατό να εισαχθούν σε κώδικα της VBA χρησιμοποιώντας τη μέθοδο **WorksheetFunction** του αντικειμένου **Application,** όπως παρακάτω:

```
Sub Excel_function()
    Dim rng As Range
    Set rng = Range("A1:A4")
    Result = Application.WorksheetFunction.Sum(rng)
    MsgBox Result
End Sub
```

# 9. Αλληλεπίδραση Εφαρμογής με το χρήστη

**Εκπαιδευτικοί Στόχοι:**

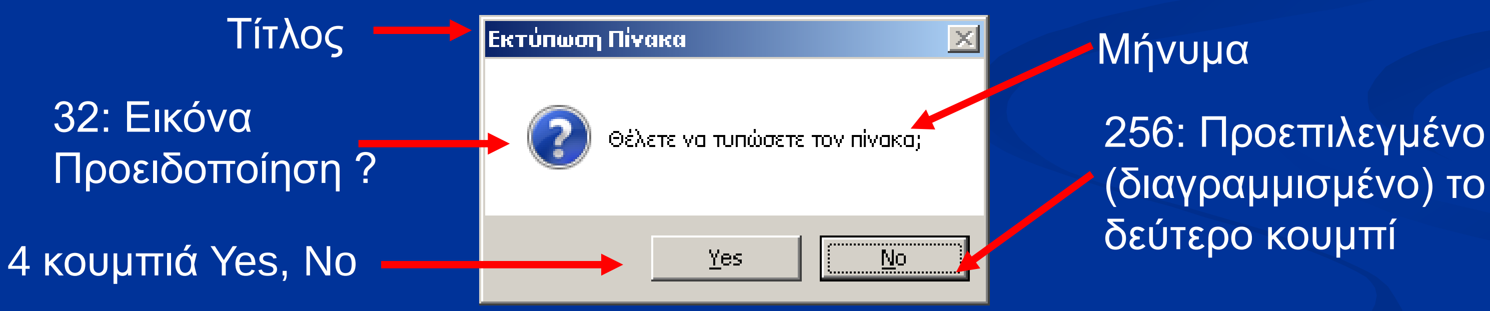
- Περιγραφή της σκοπιμότητας ύπαρξης αλληλεπίδρασης μεταξύ των εφαρμογών και του χρήστη
- Χειρισμός βασικών συναρτήσεων και αντικειμένων για τη δημιουργία αλληλεπίδρασης με το χρήστη

# Παράθυρο Διαλόγου: Συνάρτηση MsgBox

Εμφανίζει ένα μήνυμα στο παράθυρο διαλόγου, αναμένει από το χρήστη να κάνει κλικ σε ένα κουμπί και επιστρέφει μια τιμή τύπου **Integer**, η οποία δηλώνει σε ποιο κουμπί έκανε κλικ ο χρήστης.

**Σύνταξη**
**MsgBox(*ερώτημα*[, *κουμπιά*] [, *τίτλος*] [, *αρχείο_βοήθειας*, *συνάφεια*])**

**Παράδειγμα**
**Msgbox("Θέλετε να τυπώσετε τον πίνακα;";4+32+256;"Εκτύπωση Πίνακα")**



# Παράθυρο Διαλόγου: Συνάρτηση MsgBox

**MsgBox(*ερώτημα*[, *κουμπιά*] [, *τίτλος*] [, *αρχείο_βοήθειας*, *συνάφεια*])**

Η σύνταξη της συνάρτησης **MsgBox** αποτελείται από τα ακόλουθα επώνυμα ορίσματα:

| Τμήμα | Περιγραφή |
|---|---|
| ***ερώτημα*** | Απαιτείται. Παράσταση συμβολοσειράς η οποία εμφανίζεται ως μήνυμα στο παράθυρο διαλόγου. Το μέγιστο μήκος του ***ερωτήματος*** είναι περίπου 1024 χαρακτήρες, ανάλογα με το πλάτος των χαρακτήρων που χρησιμοποιούνται. Εάν το ***ερώτημα*** αποτελείται από περισσότερες από μία γραμμές, μπορείτε να διαχωρίσετε τις γραμμές με ένα χαρακτήρα επαναφοράς (**Chr(**13**)**), ένα χαρακτήρα αλλαγής γραμμής (**Chr(**10**)**) ή ένα συνδυασμό χαρακτήρων επαναφοράς–αλλαγής γραμμής (**Chr(**13**)** & **Chr(**10**)**) μεταξύ κάθε γραμμής. |
| ***κουμπιά*** | Προαιρετικό. Μια αριθμητική παράσταση που είναι το άθροισμα των τιμών που ορίζουν τον αριθμό και τον τύπο των κουμπιών που θα εμφανιστούν, το στυλ εικονιδίου που θα χρησιμοποιηθεί, την ταυτότητα του κουμπιού προεπιλογής και τον τρόπο λειτουργίας του πλαισίου μηνύματος. Εάν παραλειφθεί, η προεπιλεγμένη τιμή για το όρισμα ***κουμπιά*** είναι 0. |
| ***τίτλος*** | Προαιρετικό. Παράσταση συμβολοσειράς η οποία εμφανίζεται στη γραμμή τίτλου του παραθύρου διαλόγου. Εάν παραλείψετε τον ***τίτλο***, το όνομα της εφαρμογής τοποθετείται στη γραμμή τίτλου. |
| ***αρχείο_βοήθειας*** | Προαιρετικό. Μια παράσταση συμβολοσειράς η οποία προσδιορίζει το αρχείο της Βοήθειας που θα χρησιμοποιηθεί για να παρέχει σχετική βοήθεια για το παράθυρο διαλόγου. Εάν περιλαμβάνεται το όρισμα ***αρχείο_βοήθειας***, θα πρέπει να περιλαμβάνεται και το όρισμα ***συνάφεια***. |
| ***συνάφεια*** | Προαιρετικό. Μια αριθμητική παράσταση η οποία είναι ο αριθμός συνάφειας της Βοήθειας που έχει αντιστοιχιστεί στο κατάλληλο θέμα της Βοήθειας από το συντάκτη της. Εάν περιλαμβάνεται το όρισμα ***συνάφεια***, θα πρέπει να περιλαμβάνεται και το όρισμα |

# Παράθυρο Διαλόγου: Συνάρτηση MsgBox

**Ρυθμίσεις**
Οι ρυθμίσεις του ορίσματος ***κουμπιά*** είναι:

| **Σταθερά** | **Τιμή** | **Περιγραφή** | Ομάδα |
|---|---|---|---|
| **vbOKOnly** | 0 | Εμφάνιση μόνο του κουμπιού **OK**. | Η πρώτη ομάδα τιμών (0–5) περιγράφει τον αριθμό και τον τύπο των κουμπιών που εμφανίζονται στο παράθυρο διαλόγου. |
| **vbOKCancel** | 1 | Εμφάνιση των κουμπιών **OK** και **Άκυρο**. | |
| **vbAbortRetryIgnore** | 2 | Εμφάνιση των κουμπιών **Ματαίωση**, **Επανάληψη** και **Παράβλεψη**. | |
| **vbYesNoCancel** | 3 | Εμφάνιση των κουμπιών **Ναι**, **Όχι** και **Άκυρο**. | |
| **vbYesNo** | 4 | Εμφάνιση των κουμπιών **Ναι** και **Όχι**. | |
| **vbRetryCancel** | 5 | Εμφάνιση των κουμπιών **Επανάληψη** και **Άκυρο**. | |
| **vbCritical** | 16 | Εμφάνιση του εικονιδίου **Κρίσιμο μήνυμα**. | Η δεύτερη ομάδα (16, 32, 48, 64) περιγράφει το στυλ των εικονιδίων. |
| **vbQuestion** | 32 | Εμφάνιση του εικονιδίου **Ερώτημα προειδοποίησης**. | |
| **vbExclamation** | 48 | Εμφάνιση του εικονιδίου **Μήνυμα προειδοποίησης**. | |
| **vbInformation** | 64 | Εμφάνιση του εικονιδίου **Μήνυμα πληροφοριών**. | |
| **vbDefaultButton1** | 0 | Το πρώτο κουμπί είναι προεπιλογή. | Η τρίτη ομάδα (0, 256, 512, 768) καθορίζει πιο κουμπί είναι το |
| **vbDefaultButton2** | 256 | Το δεύτερο κουμπί είναι προεπιλογή. | |
| **vbDefaultButton3** | 512 | Το τρίτο κουμπί είναι προεπιλογή. | |



# 2.6 Παράθυρο Διαλόγου: Συνάρτηση MsgBox

Ρυθμίσεις του ορίσματος ***κουμπιά*** (συνέχεια):

| **Σταθερά** | **Τιμή** | **Περιγραφή** | Ομάδα |
|---|---|---|---|
| **vbApplicationModal** | 0 | Αποκλειστικότητα εφαρμογής. Ο χρήστης πρέπει να απαντήσει στο πλαίσιο μηνύματος πριν να είναι σε θέση να συνεχίσει να εργάζεται με την τρέχουσα εφαρμογή. | Η τέταρτη ομάδα (0, 4096) καθορίζει τον τρόπο λειτουργίας του πλαισίου μηνύματος. |
| **vbSystemModal** | 4096 | Αποκλειστικότητα συστήματος. Αναστέλλονται όλες οι εφαρμογές, μέχρι να απαντήσει ο χρήστης στο πλαίσιο μηνύματος. | |
| **vbMsgBoxHelpButton** | 16384 | Προσθέτει ένα κουμπί βοήθειας στο πλαίσιο μηνύματος | |
| **VbMsgBoxSetForeground** | 65536 | Ορίζει το παράθυρο διαλόγου του μηνύματος ως το παράθυρο στο προσκήνιο | |
| **vbMsgBoxRight** | 524288 | Το κείμενο είναι στοιχισμένο στα δεξιά | |
| **vbMsgBoxRtlReading** | 1048576 | Ορίζει ότι το κείμενο θα πρέπει να εμφανίζεται με κατεύθυνση από τα δεξιά προς τα αριστερά για το εβραϊκό και το αραβικό σύστημα | |

Όταν προσθέτετε αριθμούς για να δημιουργήσετε μια τελική τιμή για το όρισμα ***κουμπιά***, χρησιμοποιήστε έναν μόνο αριθμό για κάθε ομάδα..

**Σημείωση** Αυτές οι σταθερές καθορίζονται από τη Visual Basic for Applications. Ως αποτέλεσμα, τα ονόματα μπορούν να χρησιμοποιηθούν οπουδήποτε στον κώδικά σας αντί των πραγματικών τιμών.

# Παράθυρο Διαλόγου: Συνάρτηση MsgBox

**Τιμές που επιστρέφονται** :

| **Σταθερά** | **Τιμή** | **Περιγραφή** |
|---|---|---|
| **vbOK** | 1 | **OK** |
| **vbCancel** | 2 | **Άκυρο** |
| **vbAbort** | 3 | **Ματαίωση** |
| **vbRetry** | 4 | **Επανάληψη** |
| **vbIgnore** | 5 | **Παράβλεψη** |
| **vbYes** | 6 | **Ναι** |
| **vbNo** | 7 | **Όχι** |

**Παρατηρήσεις**

Όταν περιλαμβάνονται τα ορίσματα ***αρχείο_βοήθειας*** και ***συνάφεια***, ο χρήστης μπορεί να πατήσει το πλήκτρο F1 (Windows) ή HELP (Macintosh), για να δει το θέμα της Βοήθειας που αντιστοιχεί στο όρισμα **συνάφεια**.

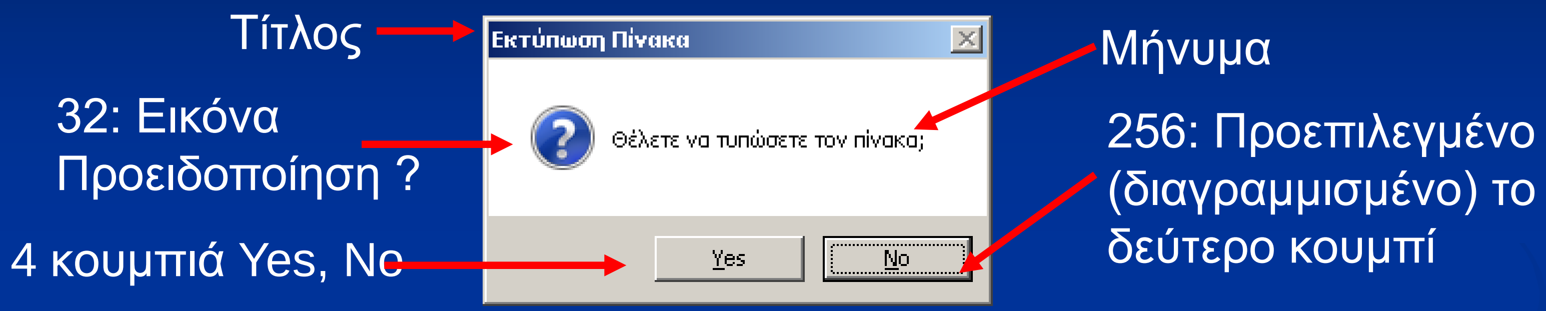
Ορισμένες εφαρμογές κεντρικού υπολογιστή, όπως για παράδειγμα η Microsoft Excel, προσθέτουν επίσης ένα κουμπί **Βοήθειας** στο παράθυρο διαλόγου.

Εάν στο παράθυρο διαλόγου εμφανίζεται κουμπί **Άκυρο**, εάν πατήσετε το πλήκτρο ESC θα έχετε το ίδιο αποτέλεσμα σαν να κάνατε κλικ στο κουμπί **Άκυρο**. Εάν το παράθυρο διαλόγου περιέχει ένα κουμπί **Βοήθειας**, παρέχεται σχετική βοήθεια για το παράθυρο διαλόγου. Ωστόσο, δεν επιστρέφεται καμία τιμή, μέχρι να κάνετε κλικ σε ένα από τα άλλα κουμπιά.

**Σημείωση** Για να ορίσετε περισσότερα από το πρώτο επώνυμο όρισμα, θα πρέπει να χρησιμοποιήσετε τη συνάρτηση **MsgBox** σε μια έκφραση. Για να παραλείψετε μερικά ορίσματα θέσης, πρέπει να συμπεριλάβετε τον αντίστοιχο οριοθέτη με κόμμα.

# Παράθυρο Διαλόγου: Συνάρτηση MsgBox

- Παράδειγμα:
  - **Msgbox("Θέλετε να τυπώσετε τον πίνακα;";4+32+256;"Εκτύπωση Πίνακα")**




Η τιμή που βάζουμε στο όρισμα Button είναι το σύνολο των τιμών των τριών πινάκων. Για παράδειγμα, αν θέλουμε να εμφανιστούν τα κουμπιά **Abort, Retry, Cancel** με την εικόνα του ερωτηματικού και να έχουμε το κουμπί **Cancel** προεπιλεγμένο τότε βάζουμε το άθροισμα: 2 + 32 + 512 ή
τον αριθμό 546

- Παράδειγμα σε μία λειτουργική μονάδα (Module):
  - **Msgbox "Θέλετε να τυπώσετε τον πίνακα;"; 4+32+256; "Εκτύπωση Πίνακα")**
- **Ασκήσεις με διάφορους συνδυασμούς**

# Ασκήσεις

- Δημιουργήστε ένα νέο Βιβλίο Εργασίας
- Γράψτε μία υπορουτίνα με όνομα PlithosGrammaton που θα ζητάει το όνομα ενός χρήστη και θα εμφανίζει σε ένα παράθυρο το μήκος των χαρακτήρων του ονόματος.

# 10. Λήψη αποφάσεων στη VBA

**Εκπαιδευτικοί Στόχοι:**

- Ανάλυση των λόγων που επιβάλουν την ύπαρξη εντολών που εκτελούν ελέγχους
- Περιγραφή των εντολών επιλογής που εμφανίζονται στη VBA και προσδιορισμός των συνθηκών κάτω από τις οποίες χρησιμοποιείται κάθε μία από αυτές
- Δημιουργία απλών εφαρμογών με τη χρήση των παραπάνω εντολών επιλογής

# Εντολή IF

- Στον προγραμματισμό μία έκφραση είναι μία ομαδοποίηση λέξεων-κλειδιών, τελεστών και/ή μεταβλητών που παράγουν ένα αποτέλεσμα σε μια μεταβλητή ή ένα αντικείμενο. Οι εκφράσεις χρησιμοποιούνται συνήθως για να πραγματοποιήσουμε υπολογισμούς και να χειριστούμε ή να ελέγξουμε δεδομένα. Επιπλέον, οι εκφράσεις μπορούν να χρησιμοποιηθούν για τη δημιουργία συνθηκών οι οποίες επιστρέφουν μια λογική τιμή Boolean TRUE ή FALSE.
- Παράδειγμα λογικής έκφρασης με συνθήκη IF

```
If arithmos1 = arithmos2 Then
    MsgBox "Ο αριθμός 1 ισούται με τον αριθμό 2"
End If
```

- Ο παραπάνω κώδικας  ελέγχει "αν η μεταβλητή aritmos1 ισούται με τη μεταβλητή arithmos2 και στη συνέχεια εμφανίζει στην οθόνη το κείμενο του Message Box. Αυτό σημαίνει ότι η έκφραση μέσα στις παρενθέσεις πρέπει να είναι αληθής, TRUE, για να εκτελεστεί ο κώδικας μέσα στο IF. Σημειώστε ότι οι παρενθέσεις γύρω από την έκφραση δεν απαιτούνται, αλλά παρέχουν αναγνωσιμότητα.
- Σημειώστε επίσης την ύπαρξη της λέξης-κλειδί Then στο τέλος της έκφρασης. Το Then απαιτείται στο τέλος κάθε έκφρασης συνθήκης IF.

# Εντολή IF

- Αλλά τι γίνεται αν η έκφραση δεν είναι αληθής, TRUE; Σε αυτή την περίπτωση έχουμε το Else. Ο κώδικας μετά το Else θα εκτελεστεί, αν η λογική έκφραση του IF είναι ψευδής FALSE.

```
If arithmos1 = arithmos2 Then
    MsgBox "Ο αριθμός 1 ισούται με τον αριθμό 2"
Else
    MsgBox "Ο αριθμός 1 δεν ισούται με τον αριθμό 2"
End If
```

- Για τις λογικές εκφράσεις στην εντολή IF μπορούμε να χρησιμοποιήσουμε τους παρακάτω τελεστές:

| **Τελεστής** | **Περιγραφή** |
|---|---|
| = | Ισούται |
| <> | Δεν ισούται |
| > | Μεγαλύτερο από |
| < | Μικρότερο από |
| >= | Μεγαλύτερο ή ίσο από |
| <= | Μικρότερο ή ίσο από |

# Εντολή IF

- Εκτός από το Else, ή VBA παρέχει και το ElseIf ως μέρος μιας μεγαλύτερης παράστασης. Το ElseIf χρησιμοποιείται για τη δημιουργία συνθηκών που ενδέχεται να έχουν περισσότερα από δύο πιθανά αποτελέσματα.

```
If arithmos1 = arithmos2 Then
    MsgBox "O αριθμός 1 ισούται με τον αριθμό 2"
ElseIf arithmos1 > arithmos2 Then
    MsgBox "O αριθμός 1 είναι μεγαλύτερος από τον αριθμό 2"
ElseIf arithmos1 < arithmos2 Then
    MsgBox "O αριθμός 1 είναι μικρότερος από τον αριθμό 2"
End If
```

- Παρατηρήστε ότι και το Elseif πρέπει να έχει τη λέξη-κλειδί Then όπως και στην έκφραση του IF
- Επιπλέον μπορούμε να χρησιμοποιήσουμε το Else ως τελική συνθήκη σε περίπτωση που καμία από τις παραπάνω συνθήκες δεν είναι αληθής.

```
If sColor = "red" Then
    MsgBox "Το χρώμα είναι κόκκινο"
ElseIf sCollor = "white" Then
    MsgBox "Το χρώμα είναι άσπρο"
ElseIf sCollor = "green" Then
    MsgBox "Το χρώμμα είναι πράσσινο"
Else
    MsgBox "Άλλο χρώμα"
End If
```

# Ένθετα IF (nested IF)

- Μερικές φορές μπορεί να χρειαστεί να έχουμε μία ή περισσότερες συνθήκες ένθετα τη μία μέσα στην άλλη. Η έννοια αυτή είναι γνωστή ως ένθετες συνθήκες και συχνά να απαιτούν πολύ προσοχή σχετικά με τη ροή του προγράμματος για την αποφυγή λαθών.
- Ο παρακάτω κώδικας έχει μία συνθήκη IF ένθετη μέσα σε μία άλλη συνθήκη IF σε ένα σύστημα απλής μισθοδοσίας:

```
If TyposMisthou = "Μηνιαίος" Then
    Pliroteo = MiniaiosMisthos
Else
    If OresErgasias <= 40 Then
        Pliroteo = KostosOras * OresErgasias
    Else
        Yperoria = (OresErgasias - 40) * KostosOras * 1.5
        Pliroteo = KostosOras * 40 + Yperoria
    End If
End If
```

- Το ένθετο IF μπλοκ εκτελείται μόνο αν η Συνθήκη στο πρώτο IF είναι FALSE. Εάν στο πρώτο ή εξωτερικό IF, η Συνθήκη είναι TRUE, τότε μισθός του υπαλλήλου είναι μηνιαίος, και η ροή του κώδικα συνεχίζει στο τελευταίο End if.
- Είναι καλό να χρησιμοποιούμε εσοχές (4, 5 κενά από την πρώτη στήλη), αντίθετα ο παραπάνω κώδικας πολύ δύσκολο να διαβαστεί.

# Σύνθετες Συνθήκες IF

- Πολλές φορές υπάρχει η ανάγκη να δημιουργήσουμε πιο σύνθετες συνθήκες στις εφαρμογές VBA. Για να δημιουργήσουμε σύνθετες εκφράσεις, χρησιμοποιούμε τους τελεστές And, Or, και Not.
- Ανεξάρτητα από τον αριθμό των τελεστών, το τελικό αποτέλεσμα μιας σύνθετης παράστασης θα είναι αληθές ή ψευδές TRUE η FALSE.
- Ο τρόπος λειτουργίας του τελεστή And φαίνεται στον παρακάτω πίνακα αλήθειας:

| Αποτέλεσμα Έκφρασης A | Τελεστής | Αποτέλεσμα Έκφρασης B | Αποτέλεσμα |
|---|---|---|---|
| TRUE | And | TRUE | TRUE |
| TRUE | And | FALSE | FALSE |
| FALSE | And | TRUE | FALSE |
| FALSE | And | FALSE | FALSE |

- Από τον πίνακα βλέπουμε ότι το αποτέλεσμα είναι TRUE μόνο αν και οι δύο εκφράσεις A και B είναι TRUE. Σε όλες τις άλλες περιπτώσεις το αποτέλεσμα είναι FALSE. Το επόμενο πρόγραμμα υλοποιεί μια σύνθετη συνθήκη χρησιμοποιώντας τον τελεστή And.

```
If (TiposMisthou = "Μηνιαίος" And Poliseis > 10000) Then
    ' Ο υπάλληλος παίρνει 20% μπόνους
    Bonus = TelikosMisthos * 0.2
    TelikosMisthos = TelikosMisthos + Bonus
End If
```

# Σύνθετες Συνθήκες IF

- Ο τρόπος λειτουργίας του τελεστή And φαίνεται στον παρακάτω πίνακα αλήθειας:

| Αποτέλεσμα Έκφρασης A | Τελεστής | Αποτέλεσμα Έκφρασης B | Αποτέλεσμα |
|---|---|---|---|
| TRUE | Or | TRUE | TRUE |
| TRUE | Or | FALSE | TRUE |
| FALSE | Or | TRUE | TRUE |
| FALSE | Or | FALSE | FALSE |

- Από τον πίνακα βλέπουμε ότι το αποτέλεσμα είναι FALSE μόνο αν και οι δύο εκφράσεις A και B είναι FALSE. Σε όλες τις άλλες περιπτώσεις το αποτέλεσμα είναι TRUE. Το επόμενο πρόγραμμα υλοποιεί μια σύνθετη συνθήκη χρησιμοποιώντας τον τελεστή Or.

```
If (Minas = "Ιούνιος" Or Minas = "Ιούλιος" Or Minas = "Αύγουστος") Then
    Epoxi = "Καλοκαίρι"
End If
```

# Σύνθετες Συνθήκες IF

- Ο τρόπος λειτουργίας του τελεστή Not φαίνεται στον παρακάτω πίνακα αλήθειας:

| Τελεστής | Έκφραση Α | Αποτέλεσμα |
|---|---|---|
| Not | TRUE | FALSE |
| Not | FALSE | TRUE |

- Από τον πίνακα βλέπουμε ότι ο τελεστής Not αντιστρέφει το αποτέλεσμα μίας πρότασης. Μία πρόταση που είναι αληθής TRUE την κάνει ψευδή FALSE και το αντίθετο.

```
If Not 5 = 5 Then
    MsgBox "ΑΛΗΘΕΣ"
Else
    MsgBox "ΨΕΥΔΕΣ"
End If
```

# Δομή SELECT CASE

- Η δομή Select Case είναι άλλο ένα εργαλείο της VBA για τη δημιουργία συνθηκών.
- Συγκεκριμένα, η δομή Select Case αξιολογεί μια έκφραση μόνο μία φορά.
- Το Select Case είναι πολύ χρήσιμο για τη σύγκριση μιας έκφρασης με πολλαπλές τιμές. Αντίθετα, ο κώδικας, με τη χρήση του IF θα ήταν πολύπλοκος και θα χρειάζονταν πολλά ένθετα IF.

```
Select Case Hmera
    Case "Δευτέρα"
        MsgBox "Εργάσιμη"
    Case "Τρίτη"
        MsgBox "Εργάσιμη"
    Case "Τετάρτη"
        MsgBox "Εργάσιμη"
    Case "Πέμπτη"
        MsgBox "Εργάσιμη"
    Case "Παρασκευή"
        MsgBox "Εργάσιμη"
    Case Else
        MsgBox "Αργία"
End Select
```

- Η δομή Select Case είναι πολύ ευέλικτη. Για παράδειγμα μπορούμε να απλοποιήσουμε την προηγούμενη δομή τοποθετώντας πολλά στοιχεία σε μια ενιαία δήλωση που διαχωρίζονται με κόμμα.

```
Select Case Hmera
    Case "Δευτέρα", "Τρίτη", "Τετάρτη", "Πέμπτη", "Παρασκευή"
        MsgBox "Εργάσιμη"
    Case Else
        MsgBox "Αργία"
End Select
```

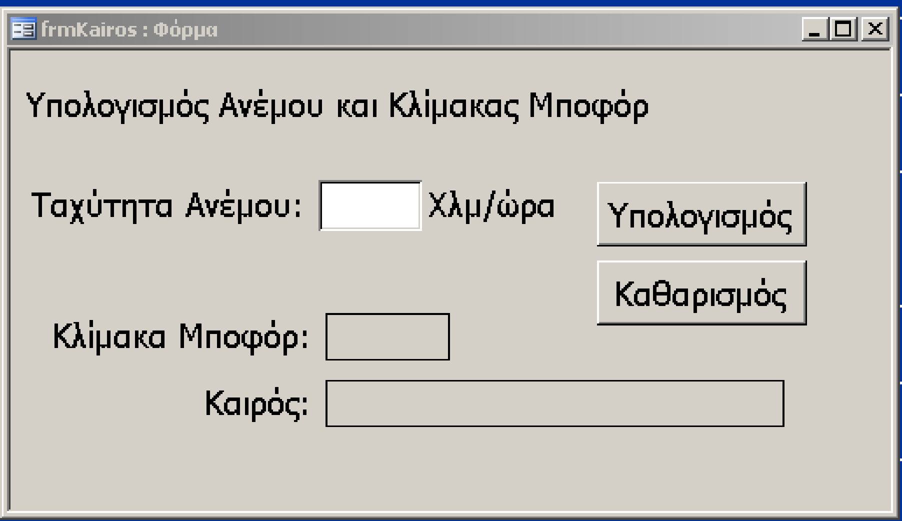
# Δομή SELECT CASE

- Η επόμενη δομή Select Case χρησιμοποιεί τις λέξεις-κλειδιά Is και To για τον έλεγχο εύρους τιμών:

```
Select Case TaxytitaAnemou
      Case Is < 1
            MsgBox "Άπνοια"
      Case 1 To 5
            MsgBox "Σχεδόν Άπνοια"
      Case 6 To 11
            MsgBox "Πολύ Ασθενής"
      Case 12 To 19
            MsgBox "Ασθενής"
      Case 20 To 28
            MsgBox "Σχεδόν Μέτριος"
      Case 29 To 38
            MsgBox "Μέτριος"
      Case 39 To 49
            MsgBox "Ισχυρός"
      Case 50 To 61
            MsgBox "Σχεδόν Θυελλώδης"
      Case 62 To 74
            MsgBox "Θυελλώδης"
      Case 75 To 88
            MsgBox "Πολύ Θυελλώδης"
      Case 89 To 102
            MsgBox "Θύελλα"
      Case 103 To 117
            MsgBox "Ισχυρή Θύελλα"
      Case Is >= 118
            MsgBox "Τυφώνας"
End Select
```

# Άσκηση

- Δημιουργήστε την παρακάτω φόρμα για τον υπολογισμό του ανέμου και τις κλίμακας μποφόρ αφού ο χρήστης εισάγει την ταχύτητα του ανέμου σε χλμ/ώρα. Η κλίμακα δίδεται στον παρακάτω πίνακα:




| Ταχύτητα (χλμ/ώρα) | | | |
|---|---|---|---|
| Από | Έως | Μποφόρ | Άνεμος |
| | <1 | 0 | άπνοια |
| 1 | 5 | 1 | σχεδόν άπνοια |
| 6 | 11 | 2 | πολύ ασθενής |
| 12 | 19 | 3 | ασθενής |
| 20 | 28 | 4 | σχεδόν μέτριος |
| 29 | 38 | 5 | μέτριος |
| 39 | 49 | 6 | ισχυρός |
| 50 | 61 | 7 | σχεδόν θυελλώδης |
| 62 | 74 | 8 | θυελλώδης |
| 75 | 88 | 9 | πολύ θυελλώδης |
| 89 | 102 | 10 | θύελλα |
| 103 | 117 | 11 | ισχυρή θύελλα |
| >=118 | | 12 | τυφώνας |

# 11. Οι Βρόγχοι στη VBA

**Εκπαιδευτικοί Στόχοι:**

- Ανάλυση των ωφελειών από τη χρήση εντολών επανάληψης στη δημιουργία εφαρμογών
- Περιγραφή των διαφορών που παρουσιάζουν οι εντολές επανάληψης μεταξύ τους και εντοπισμός των συνθηκών υπό των οποίων χρησιμοποιείται κάθε μία εξ αυτών.

# Δομές Βρόγχων (Looping Structures)

- Χρησιμοποιούμε βρόγχους σε περίπτωση που θέλουμε να επαναλάβουμε ομάδα εντολών πολλές φορές.
- Χαρακτηριστικά βρόγχων:
  - Οι βρόγχοι περιλαμβάνουν, στο εσωτερικό τους, κώδικα που επαναλαμβάνεται σύμφωνα με κάποια συνθήκη
  - Οι βρόγχοι μπορούν να έχουν ένα μετρητή
  - Οι βρόγχοι μπορούν να έχουν μία συνθήκη που αν ικανοποιηθεί ή επανάληψη τερματίζεται
  - Υπάρχουν δομές βρόγχων που έχουν τη συνθήκη επανάληψης στην αρχή ή το τέλος του βρόγχου
  - Υπάρχουν επίσης ειδικές εκφράσεις που μπορούν να τερματίσουν πρώιμα ένα βρόγχο
- Για να εξασφαλίσουμε ότι ένας βρόχους δεν είναι ατελείωτος, βάζουμε σε κάθε βρόχο μια συνθήκη που θα πρέπει να ικανοποιείται ώστε να διακοπούν οι επαναλήψεις του κώδικα. Η συνθήκη αποτιμάται σε TRUE ή FALSE.
- Παράδειγμα βρόγχου που δε σταματά ποτέ:

```
Do While 5 = 5
    MsgBox "Ατελείωτος Βρόγχος"
Loop
```

- Στη VBA έχουμε τις ακόλουθες δομές βρόγχων:
  - For
  - Do While
  - Do Until
  - Loop While
  - Loop Until

# Βρόγχος For … Next

- Στο βρόγχο For ορίζουμε ένα εύρος τιμών που καθορίζει το πλήθος επαναλήψεων του κώδικα. Κάθε φορά που εκτελείται η εντολή Next του βρόγχου ο μετρητής x αυξάνει κατά ένα.

```
Dim x As Integer
For x = 1 To 5
    MsgBox "Η τιμή του x είναι " & x
Next x
```

- Μπορούμε να ορίσουμε οποιοδήποτε εύρος τιμών επανάληψης:

```
Dim x As Integer
For x = 10 To 20
    MsgBox "Η τιμή του x είναι " & x
Next x
```

- Μπορούμε επίσης να ορίσουμε το βήμα που αυξάνει ο μετρητής:

```
Dim x As Integer
For x = 1 To 10 Step 2
    MsgBox "Η τιμή του x είναι " & x
Next x
```

- Ο κώδικας στον διπλανό βρόγχο θα εκτελεστεί 5 φορές και η τελική τιμή του x θα είναι 9.
- Το εύρος τιμών μπορεί να ορίζεται από μεταβλητές:

```
Dim x As Integer
For x = Val(Text1.Value) To Val(Text2.Value)
    MsgBox "Η τιμή του x είναι " & x
Next x
```

# Βρόγχος Do While … Loop

- Ο βρόγχος Do While χρησιμοποιεί μία συνθήκη στο πάνω μέρος του βρόγχου που καθορίζει πόσες φορές θα εκτελεστεί ο κώδικας μέσα στο βρόγχο. Επειδή η συνθήκη εκτελείται αρχικά υπάρχει πιθανότητα, αν η συνθήκη δεν ικανοποιηθεί, ο βρόγχος να μην εκτελεστεί ποτέ.

```
Dim x As Integer
Dim y As Integer

Do While x < y
    MsgBox "το x είναι μικρότερο από το y"
Loop
```

- Στο παραπάνω παράδειγμα υπάρχει πιθανότητα το x να είναι μεγαλύτερο από το y, οπότε ο βρόγχος δεν εκτελείται.
- Οι βρόγχοι Do Loop εκτελούνται όσο η συνθήκη τους είναι αληθής. Με άλλα λόγια ο βρόγχος σταματά την εκτέλεση του εσωτερικού κώδικα όταν η συνθήκη του γίνει ψευδής.
- Στο παρακάτω παράδειγμα ο βρόγχος χρησιμοποιεί μία συνθήκη που τερματίζει την εκτέλεση μετά από πέντε επαναλήψεις και η τελική τιμή του x είναι 4

```
Dim x As Integer
Dim y As Integer
x = 0
y = 5
Do While x < y
    MsgBox "Η τιμή του x είναι " & x
    x = x + 1
Loop
```

# Βρόγχος Do Until … Loop

- Ο βρόγχος Do Until, μοιάζει με το Do While, και χρησιμοποιεί συνθήκη που ορίζει πόσες φορές θα εκτελείται. Η διαφορά με το Do While είναι στον τρόπο ερώτησης της συνθήκης. Δηλαδή στην περίπτωση του Do While έχουμε τη συνθήκη: «Εκτέλεσε τον κώδικα ΟΣΟ η συνθήκη είναι ΟΡΘΗ», ενώ με την εντολή Do Until έχουμε τη συνθήκη «Εκτέλεση τον κώδικα ΜΕΧΡΙ η συνθήκη να ικανοποιηθεί»

```
Dim x As Integer
Dim y As Integer
x = 0
y = 5
Do Until x > y
    MsgBox "Η τιμή του x είναι " & x
    x = x + 1
Loop
```

- Ο παραπάνω κώδικας εκτελείται έξι φορές, και η τιμή του x μετά την τελευταία εκτέλεση θα είναι 6.

# Βρόγχος Do … Loop While

- Χρησιμοποιούμε το βρόγχο Loop While αν θέλουμε, ο κώδικας μέσα στο βρόγχο, να εκτελεστεί τουλάχιστο μία φορά ανεξάρτητα της συνθήκης του βρόγχου και στη συνέχεια να επαναλαμβάνεται η εκτέλεση του κώδικα ΟΣΟ ικανοποιείται η συνθήκη.

```
Dim x As Integer
Dim y As Integer
x = 5
y = 2
Do
    MsgBox "Σύγουρα θα εκτελεστεί μία φορά"
Loop While x < y
```

- Επειδή η συνθήκη του βρόγχου εκτελείται τελευταία, ο κώδικας μέσα στο βρόγχο, είναι σίγουρο ότι θα εκτελεστεί τουλάχιστο μία φορά. Στο τρέχον παράδειγμα θα εκτελεστεί μία μόνο φορά.

# Βρόγχος Do … Loop Until

- Χρησιμοποιούμε το βρόγχο Loop Until αν θέλουμε, ο κώδικας μέσα στο βρόγχο, να εκτελεστεί τουλάχιστο μία φορά ανεξάρτητα της συνθήκης του βρόγχου και στη συνέχεια να επαναλαμβάνεται η εκτέλεση του κώδικα ΜΕΧΡΙ να ικανοποιηθεί η συνθήκη.

```
Dim x As Integer
Dim y As Integer
x = 5
y = 2
Do
    MsgBox "Πόσες φορές θα εκτελεστεί αυτός ο κώδικας;""
    y = y + 1
Loop Until x < y
```

- Οι επαναλήψεις στο βρόγχο θα τερματίσουν όταν y πάρει την τιμή 6, δηλαδή μετά από 4 επαναλήψεις.

# Ασκήσεις

- 1. Βάλτε ένα κουμπί εντολής σε μία φόρμα. Γράψτε κώδικα που να εκτελείται με το συμβάν “Με το κλικ” όπου θα εμφανίζεται ένα πλαίσιο μηνύματος πέντε φορές χρησιμοποιώντας βρόγχο Do While. Θυμηθείτε να χρησιμοποιήσετε ένα μετρητή ως συνθήκη στο βρόγχο που θα αυξάνει κάθε φορά που εκτελείται ο κώδικας μέσα στο βρόγχο.
- 2. Τροποποιήστε την προηγούμενη άσκηση ώστε ο βρόγχος να εκτελείται 20 φορές με βήμα 3.
- 3. Επεκτείνετε την προηγούμενη άσκηση ώστε να έχουμε εκτός από τυχαίους αριθμούς και τυχαίες πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση). Χρησιμοποιήστε μία επιπλέον μεταβλητή κειμένου που θα παίρνει τιμές “ + “, “ – “, “ * “, “ / “.

# 12. Συμβάντα

**Εκπαιδευτικοί Στόχοι:**

- Ανάλυση των ωφελειών από την εφαρμογή της λογικής "driven event programming", μέσω της VBA, στη δημιουργία εφαρμογών του Excel
- Περιγραφή των διαφόρων ειδών συμβάντων

# Συμβάντα

- Συμβάντα βιβλίου εργασίας (workbook events)
- Συμβάντα φύλλου εργασίας (worksheet events)
- Συμβάντα γραφήματος (chart events)
- Συμβάντα εφαρμογής (application events)
- Συμβάντα φορμών χρήστη (user form events)
- Συμβάντα που δε συνέονται άμεσα με ένα αντικείμενο: όπως το OnKey όταν πατάμε ένα συγκεκριμένο συνδυασμό πλήκτρων στο πληκτρολόγιο.

# Αποθήκευση Δημιουργία Συμβάντων

- Ο κώδικας που θα εκτελεστεί όταν ενεργοποιηθεί ένα συμβάν αποθηκεύεται στον κώδικα του αντίστοιχου αντικειμένου που σχετίζεται το συμβάν και όχι σε κάποιο module.
- Π.χ. για να εκτελείται κώδικας όταν αλλάζει οποιαδήποτε τιμή σε οποιοδήποτε κελί του φύλλου εργασίας κάνουμε τα ακόλουθα.
  - Δεξί κλικ πάνω στην καρτέλα του φύλλου εργασίας και επιλέγουμε "Προβολή Κώδικα"
  - Πάμε κατευθείαν στον κώδικα που αφορά το φύλλο εργασίας
  - Στη αριστερή λίστα επιλέγουμε Worksheet και στη δεξιά Change
  - Δημιουργείται  αυτόματα η υπορουτίνα Worksheet_Change
  - Στη συνέχει ο κώδικας που θα γράψουμε μέσα στην υπορουτίνα θα εκτελείται κάθε φορά που κάτι αλλάζει στο εν λόγω φύλλο εργασίας

# Άσκηση

- Σε ένα φύλλο εργασίας δημιουργήστε ένα συμβάν που θα ελέγχει αν πληκτρολογήθηκε κείμενο σε κάποιο κελί. Αν συμβεί κάτι τέτοιο να εμφανίζεται μήνυμα ότι δεν επιτρέπεται η πληκτρολόγηση κειμένου και στη συνέχεια να διαγράφει το περιεχόμενο του κελιού που είναι κείμενο.

- Να γράψετε κώδικα που να εκτελείται με το άνοιγμα του βιβλίου εργασίας (Workbook_Open) που θα εμφανίζει ένα μήνυμα με τη σημερινή ημερομηνία.

# 13. Εργασία με βιβλία και φύλλα εργασίας

**Εκπαιδευτικοί Στόχοι:**

- Χειρισμός των αντικειμένων Workbook και Worksheet

# 14. Αποσφαλμάτωση

**Εκπαιδευτικοί Στόχοι:**

- Κατάδειξη της ανάγκης διαχείρισης σφαλμάτων στο πλαίσιο δημιουργίας εφαρμογών
- Δημιουργία απλών εφαρμογών στις οποίες ενσωματώνονται συναρτήσεις και αντικείμενα διαχείρισης σφαλμάτων

# Διαχείριση Σφαλμάτων

- Ο κώδικας VBA πρέπει να μεταφραστεί, μεταγλωττιστεί (compile) πριν εκτελεστεί. Η μεταγλώττιση γίνεται αυτόματα με την εκτέλεση του κώδικα.
- **Είδη σφαλμάτων**:
  - **Σφάλματα σύνταξης (Syntax Errors):** συμβαίνουν όταν παραβιάζονται οι κανόνες σύνταξης της γλώσσας VBA, για παράδειγμα:
    - Λανθασμένη δήλωση μεταβλητής
    - Ελλιπή τμήματα κώδικα που
    - Λανθασμένη πληκτρολόγηση λέξεων κλειδιών κλπ.
  - **Λογικά σφάλματα (logic Errors):** συμβαίνουν όταν ο κώδικας περιέχει λάθη που οδηγούν σε λανθασμένα αποτελέσματα. Τα λογικά σφάλματα δεν εντοπίζονται από το μεταγλωττιστή της VBA
  - **Σφάλματα μεταγλώττισης (Compile Errors):** συμβαίνουν κατά τη μεταγλώττιση του κώδικα.
  - **Σφάλματα κατά την εκτέλεση του κώδικα (Run-Time Errors):** συμβαίνουν μόνο όταν η ροή εκτέλεσης του κώδικα φτάσει στο σφάλμα, οπότε σταματάει η εκτέλεση του κώδικα και εμφανίζεται ένα μήνυμα λάθους

# Διαχείριση Σφαλμάτων

- Ακόμη και στην περίπτωση που έχουμε ελέγξει τον κώδικα, είναι πιθανό να παρουσιαστούν περιπτώσεις σφαλμάτων κατά την εκτέλεση του κώδικα σε πραγματικές συνθήκες
- Σε τέτοιες περιπτώσεις πρέπει να εξασφαλίσουμε, ότι στην περίπτωση που εμφανιστεί ένα σφάλμα, δε θα σταματήσει η εκτέλεση του κώδικα

# Παράδειγμα σφάλματος κατά την εκτέλεση του κώδικα

- Στο διπλανό παράδειγμα υπολογίζουμε το άθροισμα των κελιών A1, A2 και A3 και εισάγουμε το αποτέλεσμα στο κελί A4
- Ο κώδικας δεν περιέχει κανένα σφάλμα, αν όμως σε κάποιο από τα κελιά υπάρχει κείμενο θα εμφανιστεί Run-time error, σφάλμα κατά την εκτέλεση

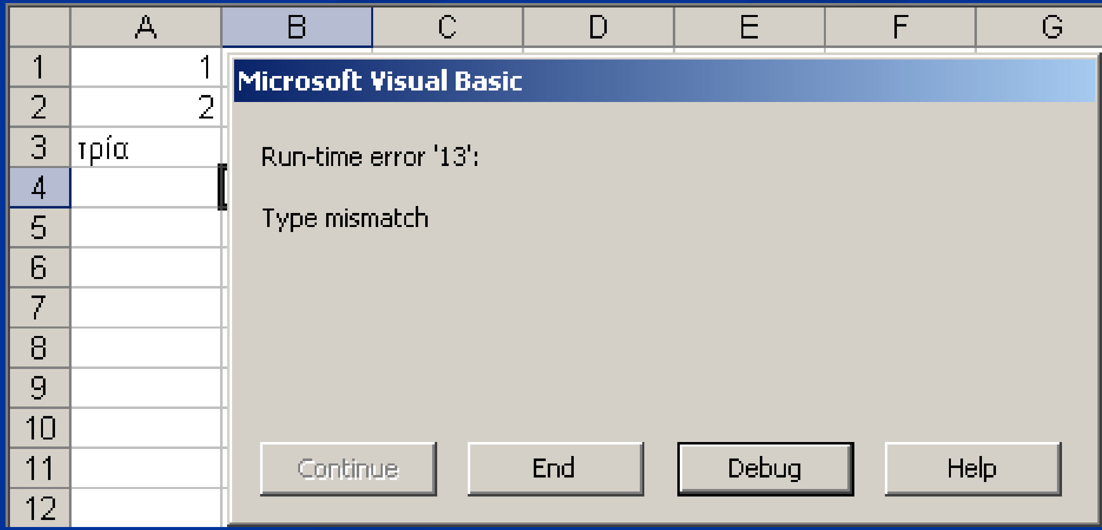
```
Sub Ignore_Error()

    Dim Total As Long
    Dim Num1, Num2, Num3 As Long

    Num1 = Cells(1, 1)
    Num2 = Cells(2, 1)
    Num3 = Cells(3, 1)

    Total = Num1 + Num2 + Num3
    Cells(4, 1) = Total

End Sub
```



# Πώς λύνουμε τέτοια προβλήματα:

1. Μετά τη δήλωση των μεταβλητών εισάγουμε στην αρχή της διαδικασίας την έκφραση **On Error GoTo Diaxeirisi_Sfalmatos**
2. Πληκτρολογούμε τον κώδικα της διαδικασίας
3. Στο τέλος του κώδικα προσθέτουμε την έκφραση **Exit Sub**
4. Μετά εισάγουμε την ετικέτα Diaxeirisi_Sfalmatos: (ή όποιο άλλο λεκτικό χρησιμοποιήσαμε μετά το GoTo) προσθέτοντας και το χαρακτήρα ":"
5. Μετά την ετικέτα πληκτρολογούμε τον κώδικα που θα εκτελεστεί αν συμβεί κάποιο σφάλμα
6. Και τέλος πληκτρολογούμε την έκφραση **Resume Next** που σημαίνει επεξεργαστής της VBA θα αγνοήσει τη γραμμή κώδικα που έχει σφάλμα και θα συνεχίζει στην επόμενη γραμμή κώδικα μέχρι να φτάσει στο **Exit Sub**

# Σχηματική Αναπαράσταση

① Αν δεν υπάρξει σφάλμα, εκτελείται ο κυρίως κώδικας και τερματίζεται η εκτέλεση με την εντολή Exit Sub

② Αν υπάρξει σφάλμα σε κάποια γραμμή κώδικα τότε η εκτέλεση του κώδικα μεταπηδά στην ετικέτα **Sfalma:**

③ Στο σημείο αυτό γίνεται προσπάθεια να διορθωθεί το σφάλμα και ο κώδικας συνεχίζει στην επόμενη γραμμή κώδικα μετά τη γραμμή που προκάλεσε το σφάλμα

④ Αν το σφάλμα δεν είναι δυνατό να διορθωθεί τότε η διαδικασία τερματίζεται χωρίς να εκτελεστεί ο κυρίως κώδικας

```
Sub MySub()
    'Δήλωση μεταβλητών
    On Error Goto Sfalma

    Γραμμή κώδικα με το σφάλμα          ②
    Επόμενη γραμμή κώδικα  <------------ ③
    ①     …
          γραμμές κώδικα
          …
    Exit Sub  ---->  Τέλος MySub()

Sfalma:
    'Κώδικας που χειρίζεται ή
    'διορθώνει το σφάλμα

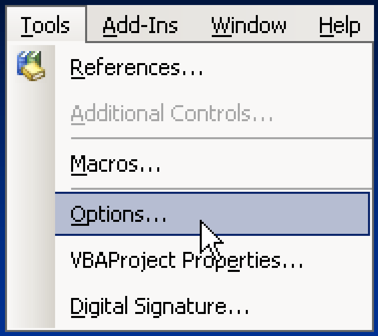
    'Αν το σφάλμα δε διορθώνεται
    Exit Sub  ------------------> ④

    Resume Next

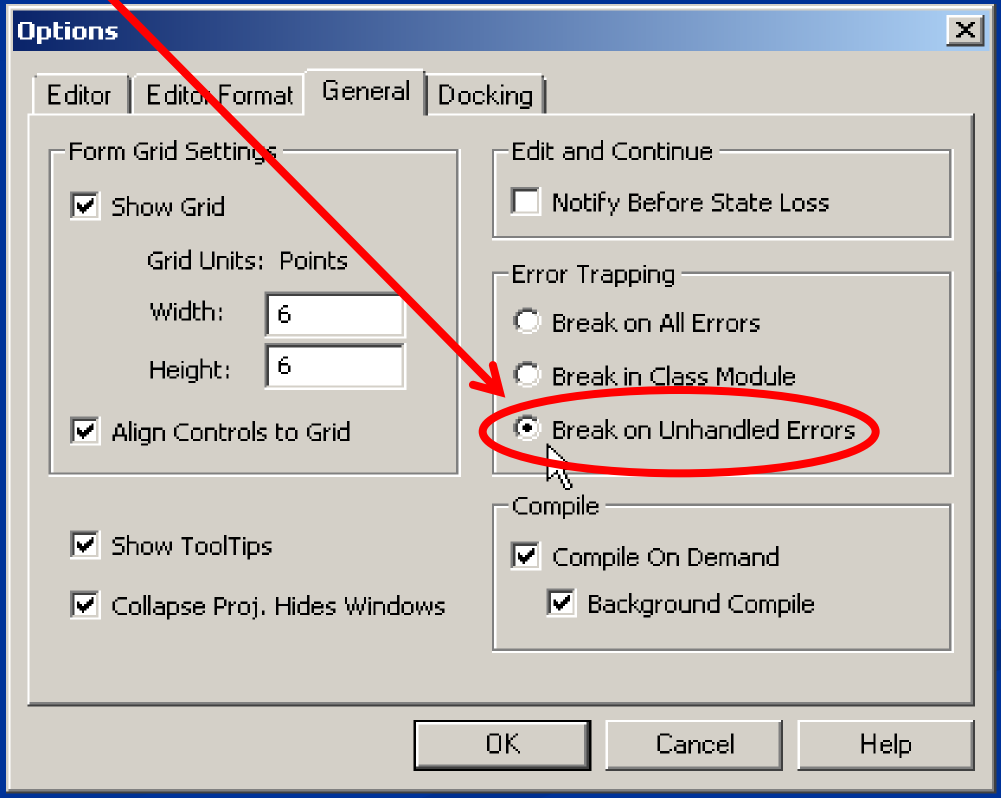
End Sub
```

# Το Παράδειγμα με διαχείριση σφάλματος

- Πρώτα βεβαιωθείτε ότι είναι ενεργοποιημένη η επιλογή **Brake on Unhandled Errors** από το μενού **Tools→Options→καρτέλα General**




- Εξ' ορισμού είναι επιλεγμένη η επιλογή **Brake on All Errors**




- Σε αυτή την περίπτωση δε θα δουλεύει η διαχείριση των σφαλμάτων με την εντολή **On Error GoTo** και το Excel θα εμφανίζει μήνυμα σφάλματος αγνοώντας τη δική μας διαχείριση σφάλματος

# Το Παράδειγμα με διαχείριση σφάλματος

```
Sub Ignore_Error()
    Dim Total As Long
    Dim Num1 As Long, Num2 As Long, Num3 As Long
    On Error GoTo Sfalma
    'Σώμα κυρίως κώδικα
    Num1 = Cells(1, 1)
    Num2 = Cells(2, 1)
    Num3 = Cells(3, 1)
    Total = Num1 + Num2 + Num3
    Cells(4, 1) = Total
    Exit Sub
'Διαχείριση Σφάλματος
Sfalma:
    If MsgBox("Εμφανίστηκε σφάλμα θέλετε να συνεχίσετε " & vbCr & _
        "αγνοώντας το σφάλμα; ", _
        vbCritical + vbYesNo, "ΠΡΟΣΟΧΗ!") = vbYes Then
        Resume Next
    Else
        MsgBox "Ο κώδικας διακόπηκε λόγω σφάλματος " & Err.Number & _
                "; " & Err.Description
        Exit Sub
    End If
End Sub
```

# Αποσφαλμάτωση: Εργαλεία και Μέθοδοι

- **Debug→Step Into ή F8:** βηματική εκτέλεση του κώδικα
- **Debug→Step Over ή Shift+F8:** Όμοια με το Step Into χωρίς όμως να εκτελεί βηματικά τις υπορουτίνες ή συναρτήσεις που καλεί ο κώδικας
- **Debug→Sep Out ή Ctrl+Shift+F8:** Εκτελεί άμεσα όλο τον υπόλοιπο κώδικα μίας καλούμενης υπορουτίνας ή συνάρτησης επιστρέφοντας στην επόμενη εντολή κώδικα από την εντολή που κάλεσε την υπορουτίνα
- **Debug→Run to Cursor ή Ctrl+F8:** Εκτελεί άμεσα όλο τον κώδικα μέχρι τη θέση του δρομέα
- **Παράθυρο Immediate**: μπορούμε να εκτελούμε άμεσα εντολές
- **Παράθυρο Locals**: Εμφανίζονται οι τιμές των μεταβλητών κατά τη βηματική εκτέλεση του κώδικα
- **Παράθυρο Watches**: Επιλέγουμε συγκεκριμένες μεταβλητές την τιμή των οποίων θέλουμε να παρακολουθούμε συνεχώς (δεξί κλικ στη μεταβλητή και Add Watch)
- **Breakpoints:** Σημεία που θέλουμε να διακόπτει ο κώδικας χωρίς να εκτελούμε βηματική εκτέλεση μέχρι να φτάσουμε στο εν λόγω σημείο (κλικ πάνω στη γραμμή κώδικα και F9 ή κλικ στη γκρίζα περιοχή αριστερά της γραμμής κώδικα)

# Άσκηση

1. Εκτελέστε βηματική εκτέλεση του προηγούμενου παραδείγματος
2. Προσθέστε Watches για τις μεταβλητές Num1, Num2, Num3 και Total
3. Βάλτε ένα Breakpoint στην εντολή κώδικα Num3 = Cell(3,1) ώστε να σταματήσετε τον κώδικα πριν εκτελεστεί το λάθος

# 15. Σύνδεση με Βάσεις Δεδομένων

**Εκπαιδευτικοί Στόχοι:**

- Διάκριση των διαφόρων ειδών σύνδεσης του Excel με Βάσεις Δεδομένων
- Υλοποίηση, μέσω καταλλήλων συναρτήσεων, σύνδεσης του Excel με την Access

# Σύνδεση με Βάσεις Δεδομένων

- Πρόσβαση σε βάσεις δεδομένων με τη βοήθεια της βιβλιοθήκης ADO (ActiveX Data Objects)
  - Σύνδεση σε οποιαδήποτε βάση δεδομένων που λειτουργεί σε περιβάλλον Windows
  - Ανάκτηση δεδομένων από τη βάση (μέσω ερωτήματος)
  - Εισαγωγή, τροποποίηση, διαγραφή δεδομένων
- Ενεργοποίηση από το μενού Tools→References της βιβλιοθήκης Microsoft ActiveX Data Objects 2.8 Library

# Παράδειγμα

```
Sub DBConnection()
    Dim cn As New ADODB.Connection
    Dim rs As New ADODB.Recordset
    Dim ws As Worksheet
    Dim sql As String
    Dim i As Integer

    Set ws = ThisWorkbook.Worksheets("Φύλλο1")
    cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source=C:\Program Files\" & _
            "Microsoft Office\OFFICE11\SAMPLES\Northwind.mdb"
    sql = "SELECT LastName, FirstName " & _
        "FROM employees ORDER BY LastName, FirstName"

    rs.Open sql, cn
    While Not rs.EOF
        i = i + 1
        Cells(i, 1) = rs!LastName
        Cells(i, 2) = rs!FirstName
        rs.MoveNext
    Wend
    rs.Close
    cn.Close
End Sub
```

# To ADO Object Model

- Η βιβλιοθήκη ADO περιέχει τα παρακάτω βασικά αντικείμενα:
  - **Connection** Δημιουργεί τη σύνδεση με τη βάση δεδομένων
  - **Command** Εντολή που περιέχει το ερώτημα (query)
  - **Parammeters** Παράμετροι για το ερώτημα
  - **Error** Μήνυμα σφάλματος
  - **Recordset** Σύνολο εγγραφών δεδομένων (data records) όπως: πίνακες, αποτελέσματα ερωτήματος κλπ
  - **Fields** Πεδία των εγγραφών δεδομένων

# Το ADO Object Model

- Τα βασικά αντικείμενα του ADO Object Model που χρησιμοποιούμε είναι τα **Connection, Command** και **Recordset**.



# Το ADO Object Model

- Τα βήματα που ακολουθούμε για να ανακτήσουμε δεδομένα από μία βάση δεδομένων, στην απλούστερη μορφή τους, είναι τα ακόλουθα:
  1. Δημιουργούμε ένα αντικείμενο **Connection** για να συνδεθούμε με τη βάση δεδομένων
  2. Ανοίγουμε το αντικείμενο **Connection**
  3. Δημιουργούμε ένα νέο αντικείμενο **Recordset**, με όλες τις αναγκαίες παραμέτρους
  4. Ανοίγουμε το αντικείμενο **Recordset**
  5. Ανακτούμε τις εγγραφές που περιέχει το **Recordset**
  6. Κλείνουμε το **Recordset**
  7. Κλείνουμε το **Connection**

# Το αντικείμενο Connection

- Δημιουργούμε τη σύνδεση με μία βάση δεδομένων, χρησιμοποιώντας τη μέθοδο **Open** του αντικειμένου **Connection**
- Μπορεί να γίνει σύνδεση με βάσης δεδομένων Access, SQL Server, Oracle, ODBC και άλλες.
- Για τη σύνδεση με την Access η έκφραση είναι:

```
Dim cn As New ADODB.Connection
cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source=C:\path\FileName.mdb"
```

- Όπου:
  - Dim cn As New ADODB.Connection δηλώνει ότι το **cn** είναι ένα αντικείμενο **Connection**
  - Provider=microsoft.jet.oledb.4.0 δηλώνει ότι η ΒΔ είναι η Access
  - Data Source=C:\path\FileName.mdb δηλώνει τη διαδρομή και το αρχείο της Access

# Το αντικείμενο Connection, Ιδιότητες

- **ConnectionString** Τρόπος σύνδεσης με Βάση Δεδομένων
- **CursorLocation** Καθορίζει αν ο τρέχον δείκτης της εγγραφής (current record pointer) θα βρίσκεται στο Excel (adUseClient) ή στη βάση δεδομένων (adUseServer). Συνήθως η δεύτερη περίπτωση είναι προτιμότερη
- **Errors** Αναφέρεται σε συλλογή αντικειμένων Error, σφαλμάτων σύνδεσης
- **Mode** Τρόπος πρόσβασης στη βάση δεδομένων
  - **adModeRead**
  - **adModeWrite**
  - **adModeShareDenyRead**
  - **adModeShareDenyWrite**
  - **adModeShareExclusive**
  - **adModeShareDenyNone**
- **State** Κατάσταση σύνδεσης (adStateClosed ή adStateOpen)

# Το αντικείμενο Connection, Μέθοδοι

- **Close** "Κλείνει" τη σύνδεση με τη βάση δεδομένων
- **Open** "Ανοίγει" τη σύνδεση με τη βάση δεδομένων

# Εκφράσεις Σύνδεσης με Βάσεις Δεδομένων

```
'Σύνδεση σε βάση δεδομένων MS Access
Dim cn As New ADODB.Connection
cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source=C:\path\FileName.mdb"

'Σύνδεση σε βάση δεδομένων MS SQL Server
Dim cn As New ADODB.Connection
cn.Open "Provider=SQLOLEDB;" & _
        "Data Source=myServerAddress;" & _
        "Initial Catalog=myDataBase;" & _
        "User ID=myUsername;" & _
        "Password=myPassword;"
```

# Εκφράσεις Σύνδεσης με Βάσεις Δεδομένων

| | |
|---|---|
| **Provider** | MSDASQL |
| **DNS** | Όνομα πηγής δεδομένων ODBC |
| **Driver** | Όνομα του οδηγού ODBC (π.χ. SQL Server) |
| **Server** | Όνομα Server (όπως εμφανίζεται στο δίκτυο) |
| **Database** | Όνομα Βάσης Δεδομένων στο Server |
| **Trusted Connection** | Yes (αν χρησιμοποιούμε Windows authentication) |
| **UID** | Όνομα χρήστη (αν δε χρησιμοποιούμε Windows authentication) |
| **PWD** | Συνθηματικό χρήστη (αν δε χρησιμοποιούμε Windows |


http://www.connectionstrings.com/
iGoogle
BitMe.ORG
TOOLS
JOB
PROGRAMMING
The Great Consumer ...
Mail Fetcher
Software Teams Worldwide
Bug Tracker · Help Desk · Agile/SCRUM
www.countersoft.com
The Connection String Reference
Database Servers
SQL Server 2008
SQL Server 2005
SQL Server Compact Edition
Oracle
MySQL
Sybase Advantage Database Server
Informix
Postgre SQL
IBM UniData
AS/400 (IBM iSeries)

# Το αντικείμενο Recordset

- Το Recordset περιέχει το σύνολο των εγγραφών που προκύπτουν όταν εκτελεστεί μία ερώτηση (query) στη βάση δεδομένων
- Χρησιμοποιούμε τη μέθοδο **Open** για να "ανοίξουμε" το Recordset: rs.Open source, activeconnection [,cursortype [, locktype]]
- Όπου:
  - **Source** Μία εντολή SQL, ή ένα αντικείμενο Command, ή το όνομα ενός πίνακα, ή ένα URL, ή μία διαδικασία (stored procedure) αποθηκευμένη στη βάση δεδομένων
  - **Activeconnection** Το αντικείμενο **Connection** που έχουμε "ανοίξει"
  - **Cursortype** (προαιρετικό) καθορίζει τον τρόπο διαχείρισης των δεδομένων από τη βιβλιοθήκη ADO και τη βάση δεδομένων
  - **Locktype** (προαιρετικό) καθορίζει αν τα δεδομένα μπορούν να τροποποιηθούν και τι θα συμβεί σε περίπτωση διένεξης, δηλ. αν παραπάνω από ένας χρήστες προσπαθήσουν ταυτόχρονα να τροποποιήσουν τα δεδομένα

# Το αντικείμενο Recordset

- Τιμές παραμέτρου **Cursortype**
  - **adLockReadOnly** Τα δεδομένα δε μπορούν να τροποποιηθούν
  - **adLockPessimistic** Κλειδώνει τα δεδομένα κατά την πρόσβαση (πριν καν γίνει προσπάθεια τροποποίησης)
  - **adLockOptimistic** Κλειδώνει τα δεδομένα κατά την τροποποίηση (μετά την πρόσβαση στα δεδομένα)
  - **adLockBatchOptimistic** Πάνω από ένα σύνολο δεδομένων μπορεί να τροποποιηθεί ταυτόχρονα

- Τιμές της παραμέτρου **Locktype**
  - **adOpenForwardOnly** Ευκολότερη διαχείριση αλλά και λιγότερες διαθέσιμες συναρτήσεις για το χειρισμό δεδομένων
  - **adOpenDynamic** Εύκολη διαχείριση με περισσότερες δυνατότητες πλοήγησης μεταξύ των δεδομένων
  - **adOpenKeyset** Δύσκολη διαχείριση και όλες οι συναρτήσεις
  - **adOpenStatic** Δυσκολότερη διαχείριση, αργή πρόσβαση στα πρώτα δεδομένα αλλά ταχύτερα τα επόμενα

# Το αντικείμενο Recordset, Ιδιότητες

| | |
|---|---|
| **ActiveConnection** | Αναφέρεται σε ένα αντικείμενο Connection, που θα χρησιμοποιηθεί για τη σύνδεση με τη βάση δεδομένων |
| **BOF** | Επιστρέφει τιμή **True** όταν ο τρέχον δείκτης εγγραφής (current record pointer) δείχνει πριν τη πρώτη εγγραφή του **Recordset** |
| **EditMode** | Δηλώνει την κατάσταση της τρέχουσας εγγραφής<br>- **adEditNone**, όταν δεν έχει τροποποιηθεί η εγγραφή<br>-**adEditInProgress**, όταν έχει γίνει τροποποίηση αλλά δεν έχει αποθηκευτεί<br>- **adEditAdd,** όταν με τη μέθοδο AddNew έχουμε δημιουργήσει μία νέα κενή εγγραφή αλλά δεν την έχουμε αποθηκεύσει<br>- **adEditDelete** |
| **EOF** | Επιστρέφει τιμή True όταν ο pointer δείχνει μετά την τελευταία εγγραφή του Recordset |
| **Fields** | Συλλογή με τα αντικείμενα **Field** του **Recordset** |
| **Filter** | Έκφραση όμοια με μία SQL δήλωση της μορφής Where… (σταθερά ή μεταβλητή τύπου string) |
| **RecordCount** | Συνολικός αριθμός εγγραφών στο Recordset |
| **Sort** | Ονόματα πεδίων που θα ταξινομηθούν οι εγγραφές |
| **Source** | Το string με την SQL ερώτηση |

# Το αντικείμενο Recordset, Μέθοδοι

| | |
|---|---|
| **AddNew** | Προσθέτει μία νέα εγγραφή μετά την τελευταία εγγραφή του Recordset |
| **CancelUpdate** | Ακυρώνει τυχόν αλλαγές της τρέχουσας εγγραφής |
| **Close** | "Κλείνει" το αντικείμενο Recordset |
| **Delete** | |
| **Move(NumRecords)** | Μετακινεί τον pointer τόσες θέσεις μπροστά ή πίσω όσο είναι η παράμετρος NumRecords |
| **MoveFirst** | Μετακινεί τον pointer στην πρώτη εγγραφή |
| **MoveLast** | Μετακινεί τον pointer στην τελευταία εγγραφή |
| **MoveNext** | Μετακινεί τον pointer στην επόμενη εγγραφή |
| **MovePrevious** | Μετακινεί τον pointer στην προηγούμενη εγγραφή |
| **Open** | "Ανοίγει" το Recordset |
| **Update** | Αποθηκεύει τις αλλαγές που έχουν γίνει στο Recordset στη βάση δεδομένων |

# Παράδειγμα: Ανάκτηση δεδομένων με Ερώτημα SQL

```
Sub CompanyByCountry()
    Dim cn As New ADODB.Connection
    Dim rs As New ADODB.Recordset
    Dim ws As Worksheet
    Dim sql As String
    Dim i As Integer

    Set ws = ThisWorkbook.Worksheets("Φύλλο1")
    cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
            "Data Source=C:\Program Files\" & _
            "Microsoft Office\OFFICE11\SAMPLES\Northwind.mdb"
    sql = "SELECT LastName, FirstName " & _
        "FROM employees ORDER BY LastName, FirstName"
    sql = "SELECT CompanyName, ContactName, Country " & _
        "FROM Customers " & _
        "WHERE Country='France' " & _
        "ORDER BY CompanyName"
    rs.Open sql, cn
    While Not rs.EOF
        i = i + 1
        Cells(i, 1) = rs!CompanyName
        Cells(i, 2) = rs!ContactName
        Cells(i, 3) = rs!Country
        rs.MoveNext
    Wend
    rs.Close
    cn.Close
End Sub
```

# Σύντομη περιγραφή της SQL (Structured Query Language)

- χρησιμοποιείται στις περισσότερες βάσεις δεδομένων
- Βασικές εντολές:
  - **SELECT** Επιλέγει ένα ή περισσότερα πεδία από ένα ή περισσότερους πίνακες
  - **INSERT** Εισάγει νέες εγγραφές σε πίνακα
  - **UPDATE** Τροποποιεί τιμές πεδίων σε εγγραφές πίνακα
  - **DELETE** Διαγράφει εγγραφές από πίνακα
  - **CREATE TABLE** Δημιουργεί νέο πίνακα
  - **DROP TABLE** Διαγράφει πίνακα
  - **WHERE** Συνθήκη περιορισμού (φίλτρο) των εγγραφών που θα προκύψουν από το ερώτημα
  - **ORDER BY** Ταξινόμηση εγγραφών ως προς επιλεγμένα πεδία
  - **GROUP BY** Ομαδοποίηση όμοιων εγγραφών ως προς πεδία που έχουν την ίδια τιμή

# Βασική σύνταξη εντολών SQL

**Ερώτημα Επιλογής**

SELECT field1 [, field2,… fieldN]
FROM table
[WHERE filter]
[GROUP BY field1 [, field2,… fieldN]]
[ORDER BY field1 [, field2,… fieldN]]

**Ερώτημα Εισαγωγής Δεδομένων**

INSERT INTO table (field1, [, field2,… fieldN])
VALUES (value1 [, value2,… valueN])
[WHERE filter]

**Ερώτημα Ενημέρωσης**

UPDATE table
SET field1=value1 [, field2=value2,… fieldN=valueN]
[WHERE filter]

**Ερώτημα Διαγραφής**

DELETE FROM table

# Παραδείγματα

Ερώτημα που επιστρέφει τα πεδία ProductName και UnitPrice από τον πίνακα Products, όπου το UnitPrice > 50

```
SELECT ProductName, UnitPrice
FROM Products
WHERE UnitPrice>50
ORDER BY ProductName
```

Ερώτημα που επιστρέφει τα πεδία ProductName, UnitPrice και CategoryName, από τους πίνακες Products και Categories, όπου UnitPrice>50

```
SELECT ProductName, CategoryName, UnitPrice
FROM Products, Categories
WHERE UnitPrice>50 AND
    Categories.CategoryID=Products.CategoryID
ORDER BY ProductName
```

# Παραδείγματα

Ερώτημα που επιστρέφει το πλήθος των εγγραφών του πίνακα Products, και τη μέση τιμή του πεδίου UnitPrice. Η λέξη κλειδί AS χρησιμοποιείται για να μετονομάσει τα πεδία που θα μπουν στο Recordset

```
SELECT  COUNT(*) AS nrOfProducts,
        AVG(UnitPrice) AS avgPrice
FROM Products
```

Ερώτημα που υπολογίζει το άθροισμα των εσόδων από τις πωλήσεις (UnitPrice * Quantity * (1 – Discount)) από τον πίνακα OrderDetails, ταξινομημένο ανά OrderID

```
SELECT  OrderID,
        SUM(UnitPrice * Quantity * (1-Discount)) AS Sales
FROM    OrderDetails
GROUP By OrderID
```

# 16. Συνεργασία με άλλες εφαρμογές του Office

**Εκπαιδευτικοί Στόχοι:**

- Δημιουργία σύνδεσης και συνεργασίας του Excel με άλλες εφαρμογές του Office

# 17. Στοχευμένη μελέτη περίπτωσης στο Δημόσιο Τομέα

**Εκπαιδευτικοί Στόχοι:**

- Εφαρμογή του συνόλου των διδαχθέντων θεμάτων μέσα από μελέτες περίπτωσης στοχευμένων στο Δημόσιο Τομέα.

# 18. Αξιολόγηση Προγράμματος

**Εκπαιδευτικοί Στόχοι:**

- Εντοπισμός των δυνατών σημείων του προγράμματος και οι ενδεχόμενες αδυναμίες ή ελλείψεις

# Αξιολόγηση Προγράμματος

- Αξιολόγηση των επιμέρους θεματικών ενοτήτων
- Αξιολόγηση της επιστημονικής επάρκειας, διδακτικής εμπειρίας και μεταδοτικότητας των εισηγητών
- Αξιολόγηση των εποπτικών μέσων, των εκπαιδευτικών υποδομών και της τεχνικής υποστήριξης
- Αξιολόγηση της πληρότητας και παρουσίασης του επιμορφωτικού υλικού

# 19. Αξιολόγηση Εκπαιδευομένων

**Εκπαιδευτικοί Στόχοι:**

- Αξιολόγηση του βαθμού αφομοίωσης των γνώσεων και δεξιοτήτων κατά την παρακολούθηση του προγράμματος

# Αξιολόγηση Εκπαιδευομένων

Τεστ – Άσκηση με:

- Ερωτήσεις πολλαπλών επιλογών ή/και
- Ερωτήματα ανοικτού τύπου ή Μελέτη Περίπτωσης